Softwarematig beleggen

Een netwerk van systemen

6-6-2012

Afstudeerscriptie

Documentinformatie

Titel	Softwarematig beleggen
Auteur	Michael Rademaker
Studentnr	2111026
Datum	06/06/2012
Versie	1

Opleiding		Organisatie	
Onderwijsinstelling	Fontys Hogenscholen ICT	Naam bedrijf	Capital Intervest
Opleidingsrichting	HBO ICT Software Engineering		
Adres	Rachelsmolen 1 R1 5612 MA Eindhoven	Adres	Bredaseweg 234 5038 NM Tilburg
Telefoonnr.	0877 877 877	Telefoonnr.	013 571 60 51
Begeleiding	Dhr. M. Franssen	Begeleiding	Dhr. F. Hoogland

Voor gezien door bedrijfsbegeleider:

Datum: 06-06-2012

Documenthistorie

Revisies

Versie	Status	Datum	Wijzigingen
0.1	Concept	20-04-2012	Opzet eerste versie.
0.2	Concept	10-05-2012	Meer paragrafen per hoofdstuk
0.3	Voorlopige versie	25-05-2012	Feedback van dhr. Hoogland en dhr. Franssen verwerkt.
1	Definitief	06-06-2012	Meer feedback verwerkt

Goedkeuring

Dit document behoeft de volgende goedkeuringen:

Versie	Datum goedkeuring	Naam	Functie	Paraaf
1	06-06-2012	Dhr. Hoogland	Bedrijf begeleider	

Distributie

Dit document is verstuurd aan:

Versie	Datum verzending	Naam	Functie
0.1	26-04-2012	Dhr. Franssen	School begeleider
0.2	15-05-2012	Dhr. Hoogland	Bedrijf begeleider
0.2	21-05-2012	Dhr. Franssen	School begeleider
0.3	27-05-2012	Dhr. Franssen	School begeleider
0.3	27-05-2012	Dhr. Hoogland	Bedrijf begeleider

Voorwoord

Na vier jaar gestudeerd te hebben, is het nu tijd om te laten zien wat ik heb geleerd en wat ik kan. Capital Intervest en Fontys geven mij deze kans.

Via Marcel Boelaars kwam ik in contact met Robbert Boutkan, directeur bij Capital Intervest. Ik raakte in gesprek met Robbert en het klikte meteen. Robbert bracht me in contact met Frank Hoogland, mijn bedrijfsbegeleider. In samenwerking met hem hebben we een stage opdracht geformuleerd en aan Fontys voorgelegd. Fontys keurde de opdracht goed waardoor ik aan de slag kon bij Capital Intervest.

Capital Intervest is een dynamische organisatie met jonge werknemers. Hier voel ik me in thuis. Ik heb altijd de werking van de financiële beurzen interessant gevonden. Wat er allemaal is, hoe het werkelijk gaat, welke regels er zijn, etc. Dit is voor mij een kans om dichter bij de beleggingswereld te komen met mijn technische achtergrond. Een kans die ik niet heb laten liggen.

Ik wil Marcel Boelaars bedanken voor het in contact brengen met Robbert Boutkan. Robbert Boutkan wil ik bedanken omdat hij mij heeft aangenomen als stagiair. Frank Hoogland wil ik bedanken voor zijn uitstekende begeleiding. Verder wil ik nog Ard van der Pijl, Bram Vermeulen, Vera Mertens en Joey de Bruyn bedanken voor hun hulp, tijdens het afstuderen. Tevens wil ik Michael Franssen hartelijk bedanken voor het begeleiden vanuit Fontys. Tenslotte wil ik Debbie Verschuren nog bedanken voor haar input voor in de scriptie.

Tilburg, juni 2012,

Michael Rademaker

Inhoudsopgave

Samenvatting

Capital Intervest is een bedrijf dat zich bezig houdt met het ontwikkelen van applicaties om een belegger te ondersteunen. Vooral het automatisch beleggen is de kracht van Capital Intervest. Beleggers zijn in staat om hun handelsstrategieën in de software te implementeren.

Mijn afstudeer periode bij Capital Intervest heeft in het teken gestaan van het koppelen van de eigen applicatie TradeRouter met een technische analysepakket. Het idee hier achter is dat de koersdata die binnenkomt in TradeRouter doorgegeven kan worden aan een technische analysepakket. Het technische analysepakket genereert een handels-signaal aan de hand van een analyse. Dit signaal pikt TradeRouter op en stuurt vervolgens een order uit naar de beurs.

Capital Intervest had haar oog laten vallen op MultiCharts. Een zeer uitgebreid technisch analysepakket. MultiCharts is zeer bekend in de professionele beleggingswereld en zou TradeRouter ook meer bekendheid kunnen geven. Tevens biedt het de klanten van Capital Intervest de kans om MultiCharts te gebruiken. Op die manier krijgen de klanten handelssignalen binnen.

Helaas kwam tijdens het implementeren van de koppeling een probleem naar boven. Er kon namelijk geen verbinding gemaakt worden tussen de koppeling en MultiCharts. Na contact te hebben opgenomen met MultiCharts Inc. werd het duidelijk dat de API die wij gebruikten niet werkte met de huidige versie van MultiCharts. Een nieuwe API zou te veel geld kosten en weinig omzet generen. We moesten dus uitwijken naar een ander technische analysepakket.

De strijd ging tussen MetaTrader en NinjaTrader. De strijd werd met een grote overwinning gewonnen door NinjaTrader. Dit simpelweg omdat MetaTrader geen API beschikbaar heeft. Verder geeft NinjaTrader zonder kosten de API mee met de software. En als laatste plus punt, is de API ook nog in dezelfde technologie geschreven als de TradeRouter.

Uiteindelijk is er een samen werking tussen Capital Intervest en NinjaTrader Inc. ontstaan. Er is nu dan ook een goedwerkende koppeling tussen NinjaTrader en TradeRouter. Klanten van Capital Intervest kunnen NinjaTrader gaan gebruiken om strategieën hierin te implementeren. Deze strategieën zullen handelsadviezen genereren en deze door sturen naar TradeRouter. Meer handelen voor de klanten dus.

Summary

Capital Intervest is a company that is specialized in developing applications that support stock traders. Especially automated trading. Traders will be able to implement their strategies in the application.

My graduation period at Capital Intervest was all about making a link between Capital Intervest's own TradeRouter with a technical analysis package. The global idea is to send stock information from TradeRouter to a technical analysis package. The package will analyse the stock information and generate a trading signal. This signal will be picked up by the TradeRouter which will, in its turn, issue an order to the stock market.

MultiCharts was the main choice of Capital Intervest. It's one of the most advanced technical analysis package available. MultiCharts is well known in the trading world and could bring TradeRouter more fame. The clients will also benefit from this link, because they can use MultiCharts to generate trading signals.

While implementing the link, there was a problem. There was no connection possible between the link and MultiCharts. After contacting MultiCharts Inc. it was made clear that the API we were using was not working with the current version of MultiCharts. A new API would cost too much money and the link wouldn't cover those expenses. We had to look for another technical analysis package.

The battle was between MetaTrader and NinjaTrader. NinjaTrader won by far this competition, because MetaTrader doesn't have an API. Plus, NinjaTrader delivers the API with the package at no charge. And above all, the API is written in the same technology as TradeRouter.

At the end, a collaboration was born between Capital Intervest and NinjaTrader Inc. This collaboration resulted in a well working link between NinjaTrader and TradeRouter. Clients from Capital Intervest can use NinjaTrader to implement strategies. These strategies will generate trading signals and send them to TradeRouter. More trading for the clients.

Verklarende woordenlijst

A

AFM	De Autoriteit Financiële Markten (AFM) houdt toezicht op de financiële markten: op sparen, beleggen, verzekeren en lenen. Het is belangrijk dat het publiek, het bedrijfsleven en de overheid vertrouwen hebben in de financiële markten. En dat de markten op een duidelijke en eerlijke manier werken. Daarom houden zij toezicht op de financiële markten.
ActiveX	Een framework ontwikkeld door Microsoft, welk in meerdere programmeer talen beschikbaar is. De verschillende talen hebben namelijk een mogelijkheid om communicatie te maken met een ActiveX component.

B

Broker	Wordt in het Nederlands ook wel effectenmakelaar genoemd. Een broker handelt in opdracht van derden. Deze derden kunnen particuliere en institutionele beleggers zijn. Een broker handelt altijd voor anderen, ze zijn namelijk strafbaar als ze voor zich zelf zouden handelen. Op iedere uitgevoerde order ontvangen zij commissie.
Beleggen	Beleggen is een vorm van investeren, vooral in financiële producten. Je geeft een bedrag op om in de toekomst het bedrag groter te maken door middel van rente of door te handelen. Dit gebeurt vaak op de beurs.
Beurs	Een fysieke of virtuele plek, waar gehandeld wordt in financiële producten.

C

C#	Wordt als C sharp uitgesproken. Het is een programmeertaal ontwikkeld door Microsoft in het .Net Framework. Het is een *managed language* die alleen op Windows draait, mits het .Net Framework is geïnstalleerd. Er zijn open source projecten die het mogelijk maken om .Net code ook op Mac OS X en Linux te draaien.

D

DLL	Dynamic Link Library. Een bestand waarin gecompileerde code te vinden is. Een dergelijk bestand kan in code worden geïmporteerd; de code kan vervolgens gebruik maken van de methodes die in het DLL bestand bevinden.

F

Framework	Een samenhang van libraries, die beschikbaar zijn voor een programmeur. Een framework bestaat dus uit code die een één of meerdere programmeurs is geschreven en gebruikt kan worden door derden. Dit bestaat vaak uit code voor bijv. wiskundige formules, zodat een programmeur die niet hoeft uit te programmeren.

K

Koers	De waarde van een effect op een bepaald moment. Koersen van effecten zijn variabel.
Koersdata	Data die het verloop van een koers in de tijd bevat.

M

Managed	Met managed wordt vaak managed code bedoeld, code dat het framework eromheen zelf voor o.a. geheugenbeheer, datatype conversie en pointers zorgt.

N

.Net Framework	Een framework ontwikkeld door Microsoft. Het is een framework dat voor managed code zorgt. Onder het .Net Framework vallen de programmeer talen C# en VB(Visual Basic).
NinjaScript	Een programmeerextensie van C#. Deze is ontwikkeld door NinjaTrader en zorgt ervoor dat je gemakkelijk kunt programmeren met NinjaTraders specifieke eigenschappen.

O

Order	Een koop- of verkoop opdracht voor financiële producten.

T

Technische analyse-pakket	Uit grafieken en koersdata kunnen door middel van formules en algoritmes, handelsadviezen worden gegenereerd. Een technische analysepakket doet dit software matig door de formules en algoritmes in te voeren. De software analyseert de data aan de hand van de opgegeven formules en/of algoritmes en genereert zo handelsadviezen.
TLB	Type Library. Het is een bestand waarin datatypes worden gedefinieerd, om zo in Visual C++ omgevingen te gebruiken.

U

Unmanaged	Met unmanaged wordt unmanaged code bedoeld. Dit houdt in dat het framework waar de code zich in bevindt, niets regelt. De programmeur moet zich bezighouden, tijdens het programmeren, met geheugenbeheer, datatype conversie etc.

1. Inleiding

Hoe maken we een koppeling tussen Capital Intervest, haar TradeRouter en een technische analysepakket? Dat is de onderzoeksvraag waar ik me mee bezig heb gehouden. Een vraag waarvan het antwoord meer mogelijkheden biedt. Want als we het antwoord hebben, dan kunnen we een koppeling gaan maken.

De TradeRouter is een applicatie om geautomatiseerd te handelen. Het is ontwikkeld door Pragmatix in opdracht van Capital Intervest. Om klanten meer te laten handelen met TradeRouter, is een koppeling vereisd met een technische analysepakket. Een technische analysepakket genereert namelijk handelsadviezen aan de hand van een analyse die uitgevoerd is op koersdata. Zo'n handelsadvies wordt dan door de TradeRouter opgepikt en uitgevoerd op de beurs.

Om tot een goede koppeling te komen, is het proces van belang. Om te beginnen is een onderzoek nodig naar de software pakketten. Nadat dit in kaart is gebracht wordt er verder onderzocht hoe deze pakketten gekoppeld kunnen worden. Nadat een methode tot stand is gekomen, worden de eisen en een ontwerp in stand gebracht. Vervolgens wordt het ontwerp gerealiseerd en getest. Tenslotte wordt de koppeling ingevoerd.

Jammer genoeg verloopt een project niet altijd zoals gepland. Dit werd duidelijk nadat er gewerkt werd met MultiCharts, het gekozen technische analysepakket. De koppeling wilde niet plaats vinden door een te oude versie van de API. Hierdoor moest er worden uitgeweken naar een ander technische analysepakket.

NinjaTrader of MetaTrader? De keuze werd al heel snel NinjaTrader, aangezien MetaTrader geen API aanbiedt. Het proces moest opnieuw worden uitgevoerd voor NinjaTrader, om tot een goede koppeling te komen. Er werd nu tijdens het onderzoeken naar NinjaTrader extra onderzocht of de API wel goed zou zijn voor de koppeling. Tijdens de realisatie en het testen is het duidelijk geworden dat NinjaTrader de beste keuze is geweest.

Na een toelichting op het bedrijf in hoofdstuk 2 en een toelichting op de opdracht in hoofdstuk 3 wordt ingegaan op de daadwerkelijke werkzaamheden. Hoofdstuk 4 vertelt u meer over de gekozen aanpak. In hoofdstuk 5 vindt u meer theorie over de beleggingswereld en in hoofdstuk 6 heb ik over de gelaagde architectuur. Verder vertelt hoofdstuk 7 hoe de koppeling tot stand is gekomen en hoofdstuk 8 beschrijft de invoering daarvan. In hoofdstuk 9 vindt u de conclusies en aanbevelingen. Tenslotte vindt u nog een persoonlijke evaluatie, de bronnen en de bijlagen.

2. Capital Intervest

2.1 De organisatie

In 2006 is Capital Intervest opgericht door Robbert Boutkan. Vanuit zijn zolderkamer heeft hij contact opgenomen met een aantal personen. Eén hiervan is Frank Hoogland. Samen met dhr. Hoogland heeft dhr. Boutkan een applicatie ontwikkeld die het handelen in aandelen automatiseert. Deze applicatie heet TradeRouter.

Pragmatix is het bedrijf achter de TradeRouter. Dhr. Hoogland is de eigenaar van Pragmatix. Capital Intervest heeft nauwe banden met Pragmatix. Er zijn een aantal afspraken gemaakt. Pragmatix ontwikkelt de software en Capital Intervest distribueert het. Eigenlijk kan worden gesproken van een joint-venture.

Capital Intervest bevindt zich in de financiële markt, in het bijzonder in de beleggingsmarkt. Capital Intervest levert software voor deze markt. Hierbij kunt u denken aan koppelingen met andere systemen, een handelsplatform en TradeRouter, om geautomatiseerd te handelen. In dat laatste heeft Capital Intervest zich gespecialiseerd.

Verder valt Capital Intervest onder het toezicht van het AFM, omdat Capital Intervest zich bezighoudt met het aanbieden van producten en diensten voor de beleggingsmarkt. Hieraan zijn wetten en regels verbonden. Om te voldoen aan deze regelgeving, huurt Capital Intervest een juridisch consultant in om er zeker van te zijn dat ze binnen de wet werken. De AFM komt om de zoveel tijd langs om te controleren of Capital Intervest zich inderdaad aan de wet houdt.

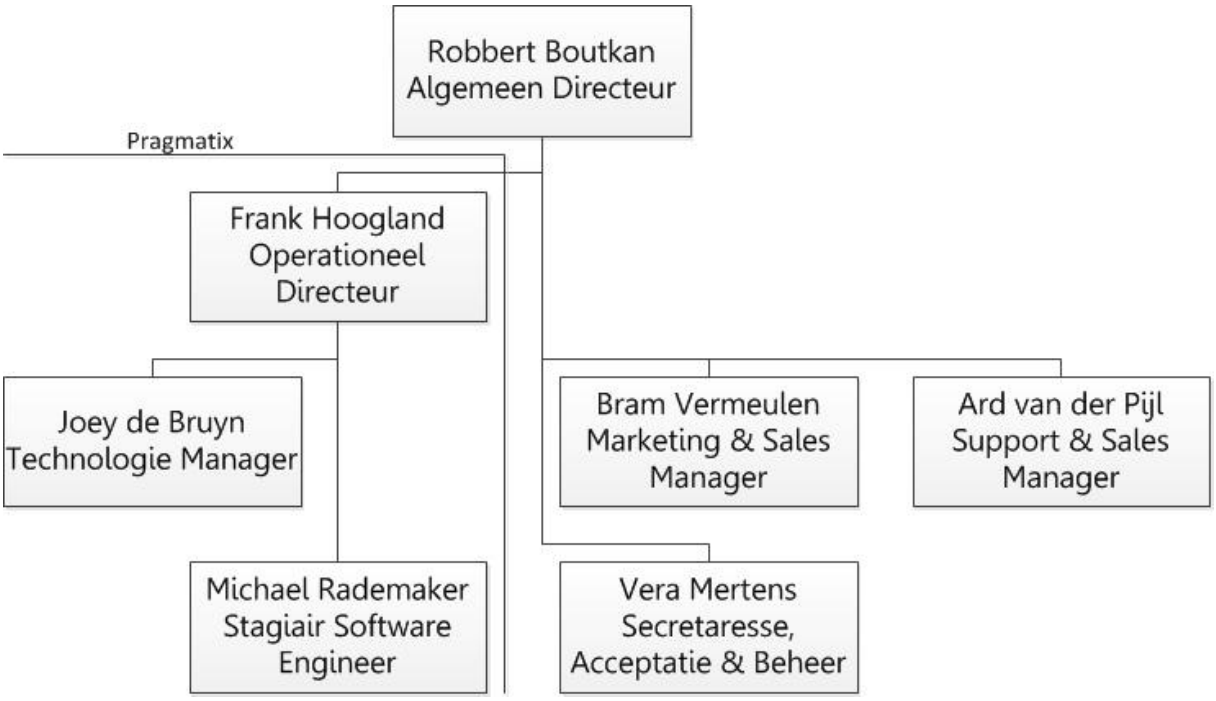
Capital Intervest werkt ook samen met brokers, onder andere IG Markets en Lynx, brokers met een eigen handelsplatform. De TradeRouter heeft hier integraties mee. Hierdoor is het mogelijk om via de software van Capital Intervest, op die platformen te handelen.

Door deze samenwerkingsverbanden aan te gaan, is voor vele beleggers TradeRouter een aantrekkelijke applicatie geworden. Ze kunnen nu namelijk geautomatiseerd handelen met strategieën. Hierdoor heeft Capital Intervest de vraag naar TradeRouter zien stijgen.

2.2 Mijn positie

Binnen Capital Intervest ben ik werkzaam bij de afdeling IT. Deze afdeling houdt zich vooral bezig met het ontwikkelen en verbeteren van software.

Bij Capital Intervest zijn er 5 werknemers en een algemeen directeur werkzaam. De afdelingen IT en Support hebben onderling veel contact, omdat de problemen die worden gemeld door klanten eerst bij Support terecht komen en daar opgelost proberen te worden. Mocht het niet lukken, dan wordt het doorgezet naar IT.



Dhr. Hoogland overlegt met Dhr. Boutkan over de roadmap van Capital Intervest. Zij samen bedenken projecten en beslissen welke projecten worden uitgevoerd. Nadat ze een beslissing hebben genomen, wordt dit bekend gemaakt aan iedereen binnen de organisatie.

Het is dus duidelijk dat ik deze twee heren aanspreek over het verloop van het project, in het bijzonder Dhr. Hoogland. Hij heeft de verantwoordelijkheid voor de IT in het bedrijf en is daarbij zeer betrokken.

2.3 Werkvloer

De gemiddelde leeftijd binnen Capital Intervest is 27 jaar. Het zijn dus jonge mensen. Dit is ook te merken aan de ideeën die het bedrijf heeft. Er wordt op technologische ontwikkelingen ingespeeld. Dit is te merken aan de constante verbetering van de software.

Er hangt een informele sfeer op de werkvloer. Het is toegestaan om een praatje met elkaar te maken en even een telefoontje te plegen. De kledingstijl is casual, maar moet wel binnen bepaalde grenzen blijven. Een voetbalshirt is bijvoorbeeld niet toegestaan, maar een pantalon is ook niet verplicht.

Ook is er tijd voor teambinding, door op vrijdagmiddag bij elkaar te gaan zitten en een drankje te nemen. Dit is ingevoerd om elkaar beter te leren kennen en meer dan alleen collegae van elkaar te zijn. Het idee hierachter is dat de samenwerking bevorderd wordt en dat bij afwezigheid, het werk van de één beter kan worden overgenomen door de ander.

3. Projectaanleiding

Na een aantal interviews met de bedrijfsbegeleider is het duidelijk geworden waarom Capital Intervest het project wil uitvoeren. Na terugkoppeling is het ook duidelijk vastgelegd wat de bedoeling is. In dit hoofdstuk komt het allemaal terug.

3.1 Koppelen met een technische analysepakket

Capital Intervest B.V. is een onafhankelijke aanbieder van producten en diensten voor de wereldwijde beleggersmarkt. Capital Intervest B.V. onderscheidt zich door een ongekende range van financiële markten en producten beschikbaar te stellen voor iedere belegger. Dit door middel van software welke exact wordt afgestemd op de wensen van de klant. Het beschikbaar stellen van die range is voor zowel de professionele belegger als de particuliere belegger.

Eén van de applicaties die Capital Intervest aanbiedt, is TradeRouter. Met TradeRouter kan de belegger, door middel van bepaalde instellingen, geautomatiseerd handelen. Dit zorgt ervoor dat de belegger niet meer achter de schermen hoeft blijven zitten en dat de computer voor hem handelt.

TradeRouter is voor het automatisch handelen afhankelijk van handelsadviezen. Er zijn al koppelingen met bepaalde technische analysepakketten. Waar echter nog geen koppeling mee is, is MultiCharts. Dit is één van de bekendste technische analysepakketten.

Er wordt gestreefd naar het aanbieden van zoveel mogelijk koersdata aan haar klanten; dit is het ideale beeld van Capital Intervest. Zoals besproken is het doel dus om alle brokers op de wereld beschikbaar te stellen aan klanten van Capital Intervest. Zo heeft de klant beschikking tot alle koersdata uit de hele wereld en is er niets onbereikbaar.

Om handelsadviezen uit al die koersdata te generen met MultiCharts, zijn er koppelingen nodig. Om dit eenvoudig te maken wordt er gestreefd om het naar één koppeling terug te brengen. Dit is het basisidee voor het project.

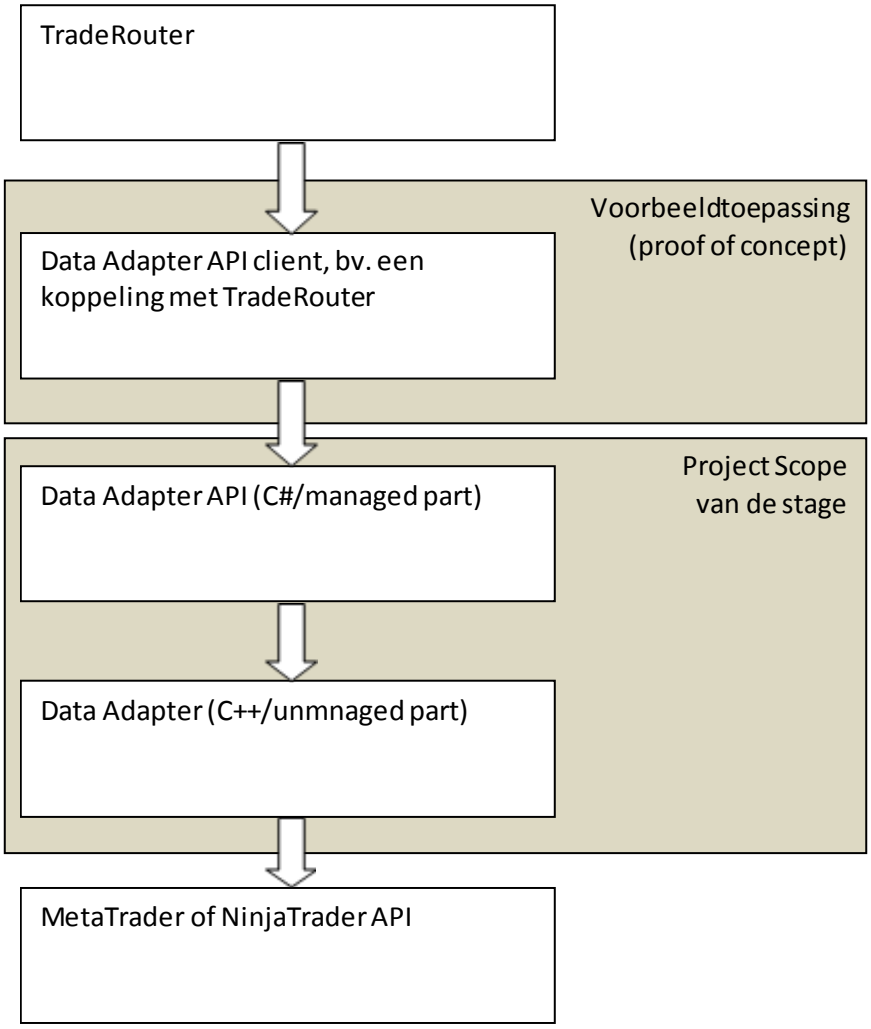
3.2 Oorspronkelijke opdracht omschrijving

Er zal worden onderzocht hoe er gekoppeld kan worden met de API van MultiCharts. Welke technologieën zijn hiervoor nodig? Hoe wordt het ingericht? Is het betrouwbaar? Werkt het efficiënt? Dit zijn allemaal vragen die beantwoord zullen worden tijdens het onderzoek.

Na het beantwoorden van de vragen zal er een ontwerp gemaakt worden. Het ontwerp zal moeten voldoen aan de eisen die later aan bod komen. Zodra het ontwerp klaar is, zal het geïmplementeerd worden. Na het implementeren volgt een testfase, en zal blijken of de conclusies van het onderzoek juist zijn.

Er wordt een ontwerp gemaakt om een universele adapter te maken. Deze adapter kan meteen door MultiCharts gebruikt worden. De koppeling naar de brokerapplicatie, in dit geval TradeRouter, zal daarna gemaakt worden. De koppeling naar TradeRouter zou voor ieder ander brokerapplicatie herschreven moeten worden. Zo kan MultiCharts makkelijk ingezet worden bij een ander brokerapplicatie.

Er is al een basisontwerp gemaakt om een idee te geven waar er naar toe moet worden gewerkt. Tevens geeft het basisontwerp aan waar naar onderzocht moet worden. Voornamelijk dat er onderzocht moet worden naar koppelingen.



Een gelaagde architectuur wordt aangemoedigd. Dit komt omdat er in de toekomst een vervanging komt voor TradeRouter. Deze vervanging zou gemakkelijk gekoppeld moeten worden met MultiCharts. Een gelaagde architectuur zou dit kunnen verwezenlijken.

3.3 Nieuwe opdrachtomschrijving

Tijdens de implementatie is naar voren gekomen dat het project toch niet zal slagen. Dit komt doordat de API van MultiCharts (die nu in bezit is) niet compatibel is met de huidige versie van MultiCharts.

Er is niet veel veranderd ten opzichte van de oorspronkelijke opdracht. De basis is nog steeds hetzelfde: Er komt een data adapter die tussen een technische analysepakket en TradeRouter verbinding legt. In de toekomst zou TradeRouter vervangen kunnen worden door een andere brokerapplicatie.

Er moet nu echter wel gekozen worden tussen twee technische analysepakketten die voorkeur hebben. Dit zijn MetaTrader en NinjaTrader. Hiervoor wordt onderzoek gedaan en uit die resultaten zal blijken welk pakket gekozen gaat worden.

3.4 Doelstelling

Wat Capital Intervest wil bereiken met het opzetten van een koppeling met een technische analysepakket, is meer mogelijkheden aanbieden aan klanten. Met een zodanig pakket kan de klant namelijk eigen strategieën uitschrijven en gebruiken in het pakket. Het pakket is dan zelf in staat om via TradeRouter een order door te geven.

Dit houdt in dat klanten meer zullen gaan handelen. Dat zorgt er weer voor dat Capital Intervest meer geld gaat verdienen. Aan iedere order die een klant uitvoert zitten namelijk transactiekosten. Deze kosten wordt door de klant betaald aan Capital Intervest.

Voor een uitgebreide omschrijving betreffende het project verwijs ik u naar het Project Initiation Document(PID) in bijlage I.

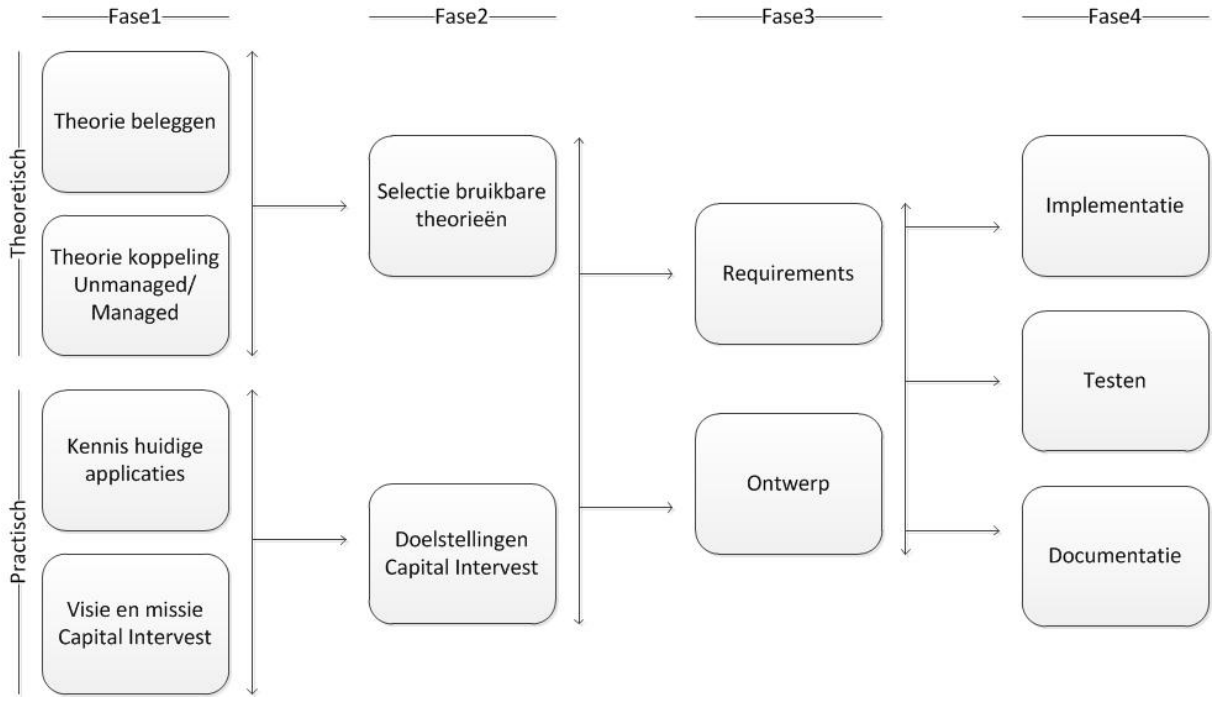
4. Aanpak

Om het project tot een succes te brengen, is het van belang dat er een goede werkplanning en projectorganisatie wordt gevormd. Dit zal helpen de afstudeerder tot een goed resultaat te leiden.

4.1 Geplande aanpak

In het project is ervoor gekozen om volgens de leer van het *Tien stappenplan* te gaan werken. Het tien stappenplan zal de leidraad zijn om het project tot een succes te brengen. Het zal ook de planning en de aanpak vorm geven. De tien stappen zullen niet allemaal in dit hoofdstuk voorkomen, maar zijn verspreid over bepaalde hoofdstukken van deze scriptie.

Ten eerste is er een start gemaakt met het definiëren van het project. Door het project te definiëren wordt het duidelijk wat de bedoeling is van het project. Tevens wordt het ook duidelijk wat de grenzen zijn van het project. Zo weten alle betrokkenen waar ze aan toe zijn. De projectdefinitie wordt opgeleverd in de vorm van een PID. In dit PID wordt meer in detail ingegaan wat op de planning is en wat de verwachtingen zijn.



Vervolgens worden er aan de hand van de planning en het PID, de volgende stappen uitgevoerd:

- Theorie over beleggen
- Mogelijke koppelingen
- Roadmap Capital Intervest
- Requirements vaststellen
- Ontwerp maken
- Implementeren, Testen en Documenteren

4.1.1 Theorie over beleggen
Theorie over hoe beleggen in zijn werk gaat is voor belang van de afstudeerder zelf. Door deze theorie te beheersen weet hij beter waar hij zich bevindt en hoe het werkt. Dit gebeurt door middel van literatuurstudie.

4.1.2 Mogelijke koppelingen
Het onderzoek van de mogelijke koppelingen van MultiCharts en TradeRouter wordt in het onderzoek uitgelegd.

4.1.3 Roadmap Capital Intervest
In de Roadmap van Capital Intervest wordt vastgelegd waar Capital Intervest zich nu bevindt. Verder wordt er uitgelicht waar ze naar toe willen met oog op het project. Tevens wordt er gesproken over de mogelijkheden die Capital Intervest heeft met de resultaten van het project.

4.1.4 Requirements vaststellen en Ontwerp
De requirements voor het product en het ontwerp daarvan komen vast te staan in het Requirements document. Aan de hand van het document kan de laatste stap worden genomen. Tevens kan de bedrijfsbegeleider nog op- of aanmerkingen maken. De uiteindelijke versie zal door de begeleider worden gecontroleerd. Het Requirements document wordt als bijlage V toevoegt voor meer detail.

4.1.5 Implementeren, Testen en Documenteren
Het Requirements document zal in deze stap werkelijkheid worden. De ontwerpen worden uit geprogrammeerd. Zodra deze gemaakt zijn en het allemaal gekoppeld is, is het tijd om te testen. Het testen is een belangrijk punt. Het gaat tenslotte om geld van klanten. Tenslotte is goede documentatie onontbeerlijk.

4.2 Onderzoek MultiCharts
De API van MultiCharts werkt met ActiveX elementen, geschreven in C++. Dit is een wat verouderde techniek, maar wordt tot de dag van vandaag nog gebruikt door veel systemen. De API van MultiCharts is er daar één van.

Omdat ActiveX in C++ ontwikkeld is, bevindt het zich in een unmanaged omgeving. Dit houdt in dat bij het programmeren, de programmeur rekening moet houden met het geheugenbeheer. Dit vraagt voor meer zorg en oplettendheid van de programmeur. Er is dus meer kans op fouten. Veel fouten zijn niet meteen te zien, maar komen pas aan het licht tijdens de uitvoering van de applicatie. Als dit het geval is, dan worden de fouten hersteld. Vervolgens moet er weer getest worden. Dit zorgt ervoor dat het proces vertraging oploopt. Daarom zou het mooi zijn als de API van MultiCharts vertaald wordt naar een managed omgeving zoals C#.

Hiervoor zijn meerdere manieren bedacht. Voor de Data Adapter moet er eerst wat onderzoek gedaan worden tussen de verschillende methodes om C# te kunnen koppelen met C++, om zo probleemloos de API te kunnen gebruiken in de .Net omgeving.

Tenslotte wordt er in kaart gebracht welke functionaliteit de API aanbiedt. Dit is te vinden in het document dat MultiCharts mee heeft geleverd: “Owndata API”.

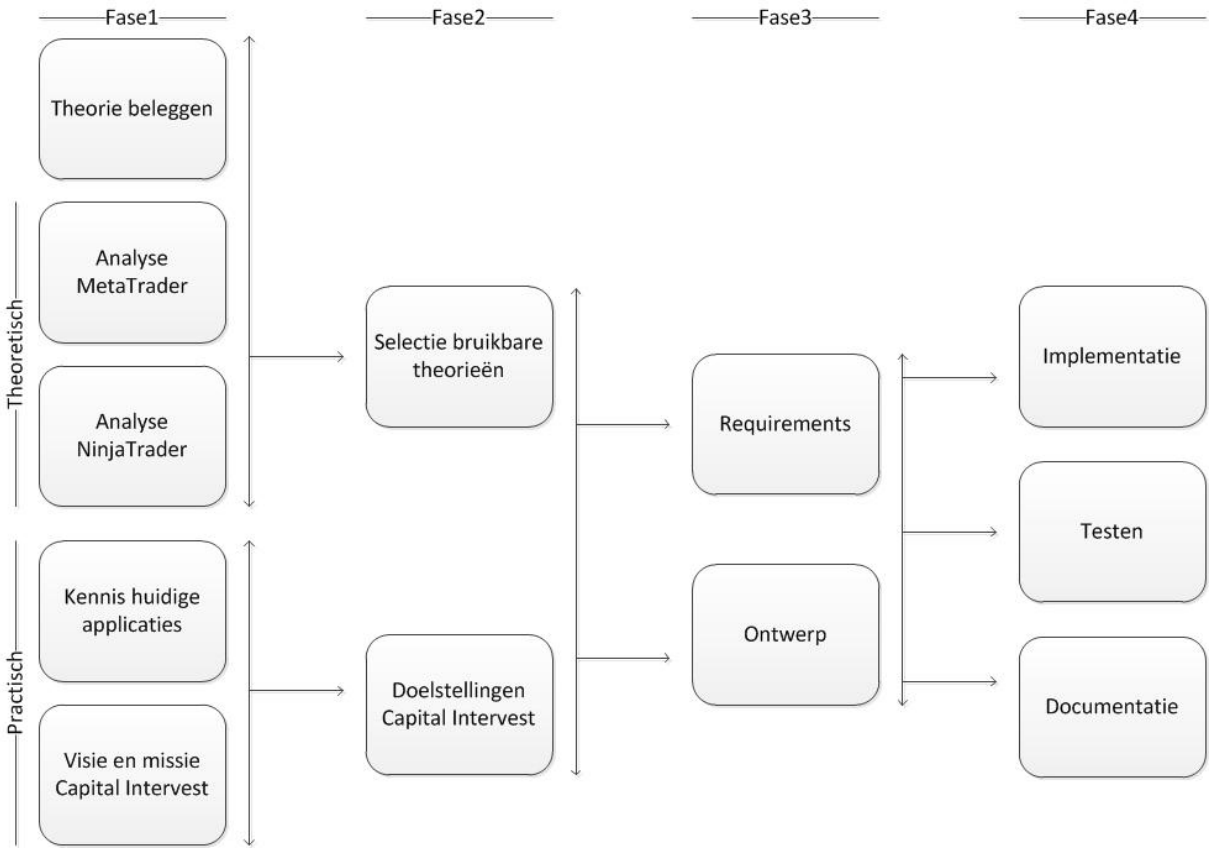
4.3 Afwijkingen ten opzichte van Planning

Tijdens het implementeren werd de zelf geschreven software door MultiCharts herkend, door middel van de API. Er werd zelfs verbinding gemaakt en het zag er naar uit dat het allemaal zou gaan lukken. Toen het moment aanbrak om prijs data over te sturen, bleef resultaat echter weg. Er werden logfiles bij gehouden om te zien wat er gebeurde. In de log files kwam naar voren dat methodes vaker aangeroepen werden dan noodzakelijk. Nadat contact was opgenomen met de support desk van MultiCharts kwam naar voren dat de API die wij in bezit hebben verouderd is.

De directie van Capital Intervest ging bespreken wat de mogelijkheden waren. Ze konden een nieuwere versie van de API aanschaffen. Maar dat bleek niet financieel niet haalbaar. De kosten van een nieuwe API lopen in de tienduizenden euro's. Het was ook duidelijk dat een dergelijk investering op redelijke termijn niet zou worden terugverdiend.

Er is namelijk geen concrete vraag om TradeRouter te koppelen aan MultiCharts. Het zou dus geen directe meerwaarde bieden aan TradeRouter als de koppeling er is. Het wordt ook voornamelijk door professionele beleggers gebruikt, en de meeste klanten zijn particulier bij Capital Intervest. Pas wanneer er veel vraag komt naar MultiCharts, zal er weer naar de situatie worden gekeken.

Om dit project toch te laten slagen is er voor gekozen om te kijken naar een alternatief voor MultiCharts. Er waren twee pakketten met voorkeur: MetaTrader en NinjaTrader. Uit onderzoek zal blijken welke van de twee het best geschikt is.



Het project verandert hierdoor minimaal. Het idee blijft hierdoor hetzelfde omdat het om 2 technische analysepakketten gaat. Het enige wat gaat veranderen is de technologie waarmee één van de alternatieve pakketten mee werkt. Er is dus ook geen herziende versie nodig van de Roadmap, aangezien het nog steeds om een technische analysepakket gaat.

Omdat het nog niet helemaal duidelijk was hoe de data-adapter eruit zou komen te zien, was het Requirements document voor de koppeling met MultiCharts nog niet af. Aangezien MultiCharts van de baan is, zal het Requirements document voor de koppeling met MultiCharts ook niet afgemaakt worden.

4.4 Onderzoek alternatief technische analysepakket

Omdat het project niet per se met MultiCharts gecombineerd dient te worden is, is het mogelijk om een andere technische analysepakket te gaan gebruiken. Er zijn er twee voor gesteld door de begeleider: MetaTrader en NinjaTrader.

Er komt een onderzoek naar beide pakketten. Er wordt een afweging gemaakt tussen de volgende punten:

- Beschikbaarheid: Is het pakket gemakkelijk te verkrijgen? Zijn er kosten verbonden aan het pakket? Hoe zit het met de support van het bedrijf?
- API: Is de API in een moderne programmeertaal geschreven? Zijn er kosten voor de API?

De uitkomst van dit onderzoek zal bepalen welk technische analysepakket MultiCharts gaat vervangen. Er wordt dan een universele data-adapter geschreven voor het gekozen technische analysepakket. Omdat het tijdens het project snel voortgezet moet worden en er alleen een pakket gewijzigd wordt, is er geen nieuwe PID nodig.

5. De beleggingswereld

5.1 Een stukje theorie

De beleggingswereld is onderdeel van de financiële markt. Onder de beleggingswereld zijn er vele vormen van investeringen. Hierbij kan men denken aan aandelen, opties, futures, maar ook in vastgoed kan men investeren.

Er zijn dus vele financiële producten, ieder met zijn voor- en nadelen. De één biedt meer zekerheid, maar minder rendement. Hier kun je denken aan vastgoed. Als u een hypotheek neemt en een huis koopt, dan belegt u in uw eigen huis. Een huis zal vaker in waarde stijgen dan dalen, dus biedt het meer zekerheid. Het zal echter niet *snel* stijgen, waardoor het rendement laag blijft.

Aandelen daarentegen bieden meer rendement, dus zullen sneller stijgen van de waarde. Maar sneller stijgen betekent ook dat het sneller kan dalen. Dit brengt dus meer risico met zich mee. Het is dus van belang dat u zich eerst wat meer verdiept in een bedrijf voordat u daar een aandeel van koopt.

Redenen om te gaan beleggen is dat het meer rendement oplevert dan sparen. Bij sparen ligt de rente laag, vaak tussen de 2 en 4 procent. Als u in aandelen of opties gaat handelen, dan kunt u al snel een rendement van 20% krijgen. Men gaat dus beleggen om hun geld sneller te laten groeien dan het geval is bij sparen.

Beleggen gebeurt via banken of brokers. Deze instanties zijn bevoegd om voor klanten orders te mogen uitvoeren op de beurs. Een belegger kan dan via een telefoontje of via een applicatie een order doorgeven. De bank of broker pakt dit op en voert het op de beurs uit. Als je dus wilt beleggen, dan dien je een rekening te openen bij een bank of een broker. Met deze rekening kun je dan gaan beleggen.

Omdat het hier om geld gaat van de consument, zijn er ook regels aan verbonden. De Nederlandsche Bank en de Autoriteit Financiële Markten (AFM) houden toezicht op de financiële markt. Ze controleren vermogensbeheerders, banken, brokers, adviesbureaus en vele andere instellingen die te maken hebben met financiële producten.

De AFM is hier actiever in en gaat bij bedrijven langs om de administratie te controleren. Ze kijken of er niet gefraudeerd wordt en of alle gegevens en registraties minstens vijf jaar bewaard worden. Verder kijken ze na of de instantie ook een workflow heeft voor aanmeldingen van nieuwe klanten en een workflow voor klachten. Deze moeten voldoen aan de richtlijnen van de AFM.

Vergunningen uitdelen is ook een taak die bij de AFM hoort. Een instelling kan bijvoorbeeld aanvraag doen voor een vergunning om beleggingsadviezen te mogen verstrekken. De AFM komt dan langs om dit te bespreken en te kijken of de instelling aan de eisen voldoet. Mocht dit niet zo zijn, dan moet de instantie verder werken om haar bedrijfsvoering zodanig te wijzigen om wel aan de eisen te voldoen.

5.2 Waar bevindt Capital Intervest zich

Capital Intervest is nu nog een introducing broker. Dat houdt in dat dat Capital Intervest orders van klanten mag aannemen en die doorgeeft aan een bank of broker. Capital Intervest voert deze dus niet uit, want anders zouden ze strafbaar zijn. Ze hebben namelijk geen vergunning hiervoor.

De AFM is tijdens de afstudeerperiode langs geweest om te controleren of Capital Intervest in aanmerking komt voor een brokervergunning. Ze kwamen kijken wat TradeRouter inhield en of het niet meer deed dan toegestaan was. Bovendien heeft de AFM naar de opslag van klantgegevens en klachten gekeken of dit in orde is. Tenslotte hebben ze nog bekeken of Capital Intervest een klachten- en aanmelding workflow heeft die voldoet aan de richtlijnen.

Tenslotte dient Capital Intervest voor een broker vergunning een AO/IC(Administratieve Organisatie / Interne Controle) handboek te hebben. Hierin staan de processen beschreven over hoe klantgegevens worden opgeslagen. Ook worden hier vragen en klacht afhandelingen beschreven. Bovendien wordt er beschreven hoe de controle intern plaats vindt. Tot slot staat erin wie de eigen(a)ar(en) is/zijn van Capital Intervest.

Na een demonstratie van TradeRouter en een paar besprekingen heeft de AFM een broker vergunning verstrekt aan Capital Intervest. Er dienen nog wel een paar aanpassingen gedaan te worden aan de workflows. Ze komen nog een keer langs om te kijken of deze wijzigen zijn doorgevoerd. Als deze wijzigingen zijn doorgevoerd, dan hoeft Capital Intervest zich geen zorgen te maken en mag zij de brokervergunning behouden.

Capital Intervest is dus bezig met de transitie om orders te gaan uitvoeren. Zodra dit plaats heeft gevonden, is Capital Intervest officieel een broker.

6. Gelaagde architectuur

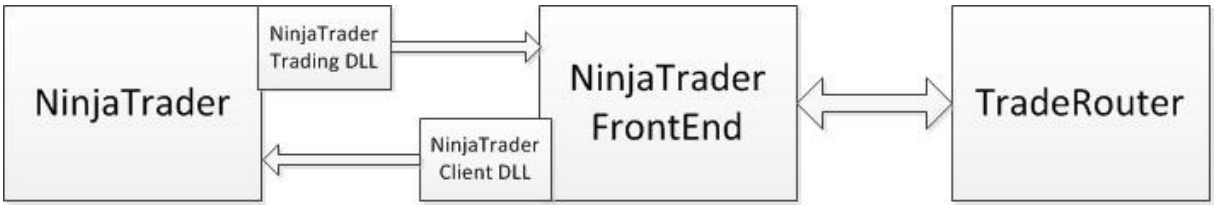
6.1 Eigenschappen

Tegenwoordig wordt software vaak in lagen geschreven. Deze vorm van software architectuur wordt ook wel gelaagde architectuur genoemd. Een veel voorkomende vorm van deze architectuur is een 3-lagenachitectuur. Hierin is er een afzonderlijke laag voor de presentatie, business logica en datatoegang/dataopslag. Het is mogelijk om nog meer lagen in de architectuur te werken, om zo iedere laag een specifieke rol te geven.

Doordat de software is opgedeeld in lagen, is het ook weer mogelijk om functionaliteit te ontkoppelen en te vervangen door een andere laag in te zetten. De lagen worden aan elkaar gekoppeld en werken zo als een geheel.

Nog een eigenschap van een gelaagde architectuur is dat wijzigingen die in een bepaalde laag plaatsvinden, de andere lagen niet beïnvloeden. Dit maakt het opsporen van fouten ook eenvoudiger, omdat de programmeur weet in welke laag hij of zij als laatste aanpassingen heeft verricht.

Tevens kan een laag worden hergebruikt in een ander systeem, waardoor er niets herschreven te worden. Een laag heeft dus zijn eigen verantwoordelijkheden en kan onafhankelijk ingezet worden in meerdere systemen.



Gelaagde architectuur in het project.

6.2 Voordelen

Er zijn vele voordelen te vinden in de gelaagde architectuur. Structuur, geordendheid en overzicht zijn drie sterk voorkomende aspecten in de gelaagde architectuur. Daarom is dit ook een veel gebruikte architectuur die vrijwel onmisbaar is.

Zo is het duidelijk waar je code moet plaatsen voor bepaalde handelingen. Ook wordt het zoeken naar fouten makkelijker, omdat wanneer er een fout optreed in bijv. het opslaan van een gegeven in de database, dat je dan in de data-laag naar de fout moet gaan zoeken.

Dus op extendibility en maintainability scoort deze architectuur hoog, omdat het gemakkelijk is om uit te breiden. Je weet ten slotte waar je je uitbreiding kwijt moet. Voor het opzoeken van fouten of het uitvoeren van optimalisatie weet je ook waar je moet zijn vanwege de scheidingen van functionaliteit.

Deze architectuur laat ook zien waar bepaalde intelligentie thuis hoort. Wanneer er zeer specifieke taken moeten worden uitgevoerd, zoals het juiste commando terugsturen, dan doe je in de onderste laag. De lagen er boven zijn abstract zodat er zeer generiek kan worden gezocht naar het/de juiste commando(s). Met andere worden: Hoer hoe hoger in de

lagen, hoe abstracter en meestal intelligenter de code is, hoe lager je komt, hoe specifieker (meer hard coded) de code is.

De programmeurs kunnen zich ook specialiseren in een bepaalde laag, waardoor ze gemakkelijk fouten kunnen opsporen in die bepaalde laag. Met deze specialisatie kunnen ze het uiterste halen uit iedere laag en ook goed gedocumenteerde informatie vast leggen.

Werk kan parallel worden uitgevoerd, zodat iedereen in een laag kan werken. Dit zorgt ervoor dat er meer werk verzet kan worden in dezelfde tijd. Vooral omdat er niet tegelijk aan hetzelfde wordt gewerkt en zo dus werk kan worden overschreven.

6.3 Nadelen

Toch zit er een nadeel aan deze architectuur. Voor het uitvoeren van een bepaalde taak kan het zijn dat er onnodig werk verricht moet worden. Het idee van deze architectuur is dat alles via de lagen door wordt gegeven. Dat houdt in dat een verzoek van de ene laag naar de ander gaat. Nu gaat dit niet langs alle lagen, maar wel door een aantal.

Dit zorgt ervoor dat er vaak een verzoek naar een laag moet die het alleen maar doorgeeft aan een andere, zonder er zelf iets mee te doen. Hierdoor kan performance wat achteruit gaan en wordt het geheel wat inefficiënt.

7. Ontstaan van een eindproduct

Met de resultaten van het diepteonderzoek is het tijd om het oplossingsplan uit te voeren. De resultaten van het diepteonderzoek zullen worden besproken. De stappen naar het oplossingsplan zullen ook worden behandeld.

7.1 MultiCharts Data adapter API

Ondanks dat er niet verder met MultiCharts wordt gewerkt, is het interessant om toch het proces door te nemen. Er zijn namelijk resultaten gevonden en keuzes gemaakt. Ook geeft het informatie over hoe MultiCharts gekoppeld zou kunnen worden via C#.

7.1.1 Resultaten van het onderzoek

Uit het onderzoek heeft er een methode voorkeur gekregen en wordt er daarop ontwikkeld. Deze methode bestaat uit het aanbieden van Interfaces. Zo kun je vanuit twee lagen, ongeacht de technologie, aanroepen uitvoeren. Dit gebeurt door middel door van het genereren van een TLB bestand uit het C# gedeelte. Dit TLB bestand kan dan weer worden geïmporteerd in een C++ omgeving. Via de TLB kan er een instantie gemaakt worden van een object uit het C# gedeelte.

```
#import "..\CsharpPart\bin\Release\CsharpPart.tlb" raw_interfaces_only
using namespace CsharpPart;

class ManagedDLLInterface
{
protected:
        IManagedMultiChartsAPIPtr pIMC;
        void InitInterface()
        {
                if (pIMC == NULL)
                {

        pIMC = new IManagedMultiChartsAPIPtr( __uuidof(ManagedMultiChartsAPI));
                }
        }
…
```

De variabele pIMC is een instantie van een ManagedMultiChartsAPI. Dit is de API van MultiCharts geschreven in C#. MultiCharts doet een aanroep op de methode get_Name(). De MangedDLLInterface class doet vervolgens een aanroep op getName() van ManagedMultiChartsAPI. ManagedMultiChartsAPI implementeert uiteindelijk de methode en geeft in dit geval een string terug. Deze wordt opgepikt door MangedDLLInterface en geeft het weer door aan MultiCharts.

Voor de andere methodes om Unmanaged C++ te koppelen met Managed C#, verwijs ik u naar het document “Unmanaged naar Managed”. Deze wordt als bijlage III toegevoegd.

7.1.2 Voorbereiding

Om fouten in het ontwerp te voorkomen is het noodzakelijk om voor te bereiden. Het zal tijd besparen tijdens de implementatie, omdat er nagedacht is over hoe het product eruit moet komen te zien. Het helpt om de lagen nader te bestuderen en goed uit te schrijven, zodat het

duidelijk wordt hoe deze moeten worden geïmplementeerd. Dit allemaal wordt vastgelegd in het Requirements document.

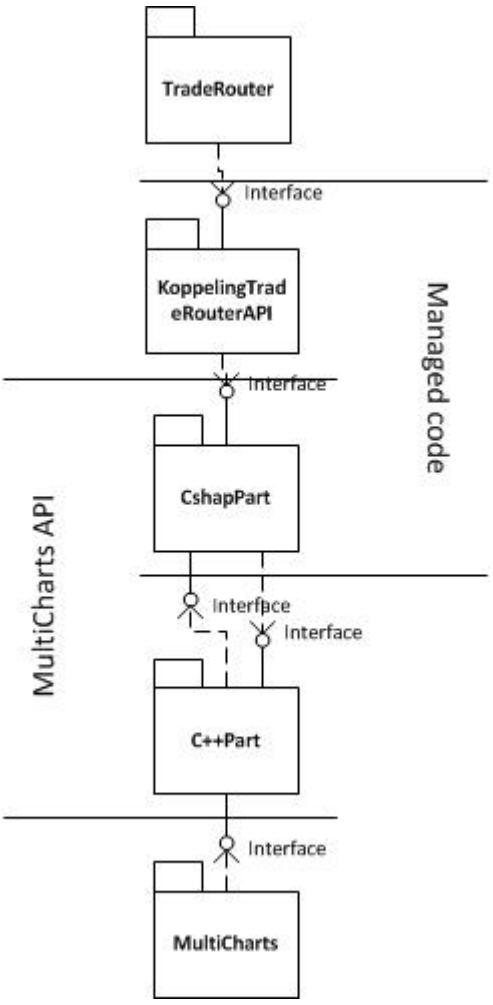
Het koppelen van TradeRouter met MultiCharts gebeurd door middel van het maken van lagen in de software en die met elkaar te verbinden. Via deze lagen kan TradeRouter koersdata leveren aan MultiCharts. Dit is in grote lijnen de opzet van de software. Er zijn drie lagen aanwezig in het project:

- MultiCharts OwnData API C++
- MultiCharts API C#
- Koppeling tussen TradeRouter en MultiCharts API in C#

In het Requirements document wordt er per laag uitgelegd wat zijn rol is in het geheel, wat de kenmerken zijn en hoe het gebruikt wordt. Per laag wordt er behandeld:

- Genomen beslissingen met onderbouwende argumenten.
- De eisen en wat ermee wordt bedoeld
- Ontwerpen
- Voorbeeldscenario's

Zoals te zien is aan de afbeelding hiernaast, zal er per laag een interface worden aangeboden. Via deze interfaces kunnen de lagen met elkaar communiceren. Door het op deze manier te ontwikkelen, is het mogelijk om de lagen onafhankelijk van elkaar te ontwikkelen en te gebruiken.



7.1.3 Realisatie

De OwnData API van MultiCharts kwam al aangeleverd en hoefde alleen nog maar in gebruik worden genomen. Voor Van der Moolen BV(een broker) was dit al gedaan dus kan het Unmanaged gedeelte meteen worden overgenomen.

Voor Van der Moolen zelf was ook een managed API gemaakt, alleen was deze geschreven in Visual Basic. Het C# gedeelte hoefde alleen maar vertaald te worden vanuit Visual Basic. Het was dus een snelle en simpele klus om het C# gedeelte te schrijven.

Vervolgens dient er een TLB bestand te worden gegeneerd. Deze wordt met behulp van een DLL gemaakt. De TLB wordt dan vervolgens in het C++ gedeelte geïmporteerd. Via een TLB kan de C++ code de classes in het C# gedeelte herkennen. Zo is het gemakkelijk om de C# methodes aan te roepen in C++.

```
// Import the type library.
#import "..\ManagedDLL\bin\Release\ManagedDLL.tlb" raw_interfaces_only

using namespace ManagedDLL;

class ManagedDLLInterface
{
protected:
        IManagedClassPtr pIMC;

        void InitInterface()
        {
                if (pIMC == NULL)
                {
                        pIMC = new IManagedClassPtr( __uuidof(ManagedClass));
                }
        }
public:
        ManagedDLLInterface()
        {
                // Initialize COM.
                HRESULT hr = CoInitialize(NULL);
                pIMC = NULL;
        }

        ~ManagedDLLInterface()
        {
                pIMC = NULL;
                // Uninitialize COM.
                CoUninitialize();
        }

        void DoSetParent(ICVDMSupplier *obj)
        {
                InitInterface();
                pIMC->SetParent(obj);
        }

        BSTR get_Name()
        {
                BSTR strResult;
                InitInterface();
                pIMC->get_Name(&strResult);
                return strResult;
        }
…
```

Om te testen of het C# gedeelte verbinding kan maken met MultiCharts, is er een applicatie gemaakt. Deze applicatie genereert een fictieve koers die opgevraagd kan worden. Via het C# gedeelte wordt de koersdata overgestuurd.

7.1.4 Tegenslagen

De realisatie van het product is van korte duur geweest. Dit vanwege compatibiliteitsproblemen van de OwnData API die Capital Intervest in bezit heeft met de meest recentste versie van MultiCharts.

In MultiCharts was de data adapter beschikbaar. Er kon verbinding worden gemaakt, maar niet meer dan dat. In de log files was ook te zien dat bepaalde methodes vaker werden aangeroepen dan nodig bleek te zijn. Ook werden er willekeurig methodes aangeroepen zonder aanleiding. Dit gaf reden om nog eens navraag te doen bij MultiCharts hoe het zat met de API.

Nadat er contact was gelegd met de support desk van MultiCharts, was het duidelijk geworden dat we een verouderde API hadden. MultiCharts heeft een ander architectuur gekregen in de nieuwere versie en kon niet meer overweg met onze API.

Een nieuwe API moest worden aangeschaft. Helaas kost een nieuwere API veel geld. Meer geld dan het zou gaan opleveren. Dit blijkt uit cijfers van Capital Intervest. Er is maar een kleine groep klanten die MultiCharts kent en een nog kleinere groep die gebruik maakt van MultiCharts. Een koppeling maken met MultiCharts zou haast geen nieuwe klanten trekken. Het zou dus geen winst opleveren voor Capital Intervest om een nieuwe API aan te schaffen.

Vandaar dat een alternatief moet worden gevonden.

7.2 FrontEnd

7.2.1 Resultaten van het onderzoek

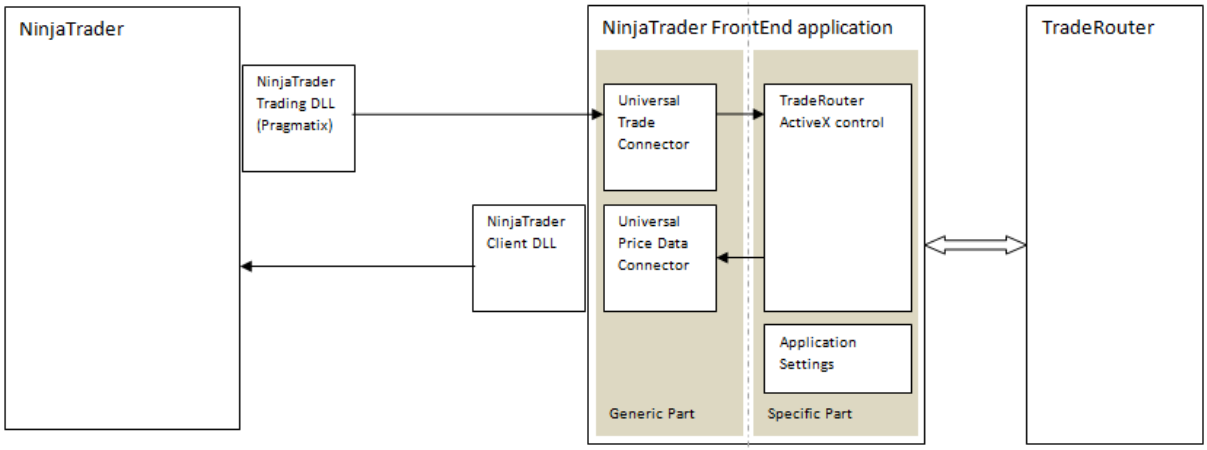
Uit het onderzoek naar een alternatieve technische analysepakket is al snel gebleken dat NinjaTrader de keuze is. Dit komt omdat er voor MetaTrader geen officiële API beschikbaar is.

Tevens wordt de API van NinjaTrader meegeleverd met het pakket zelf. Dit zorgt ervoor dat er kosteloos gebruikt kan worden gemaakt van de API. Er hoeft dus ook geen moeite gedaan te worden voor het verkrijgen van de API.

Het laatste pluspunt is dat de API van NinjaTrader geschreven is in C#. Een speciale tussen laag voor Unmanaged en Managed vertaling is niet meer nodig. De API is beschikbaar als een DLL. Deze DLL kan geïmporteerd worden in een C# omgeving.

Er is echter wel een nadeel aan de API van NinjaTrader. Deze geeft geen mogelijkheid om een handelssignaal terug te geven aan een brokerapplicatie. Hiervoor moet dus een omweg worden ontwikkeld. Deze omweg wordt gevonden in NinjaScript.

NinjaScript is een extensie van C#, waarmee een strategie in code wordt geschreven. Deze strategie wordt vervolgens losgelaten op een koers in NinjaTrader. Omdat NinjaScript gebaseerd is op C#, is het mogelijk om in een script een DLL in te laden. Via een DLL is het mogelijk om een signaal te sturen naar de brokerapplicatie. Hierdoor wijzigt het basis ontwerp enigszins.



In het document "Een alternatief technische analysepakket" wordt het onderzoek verder uitgelicht.

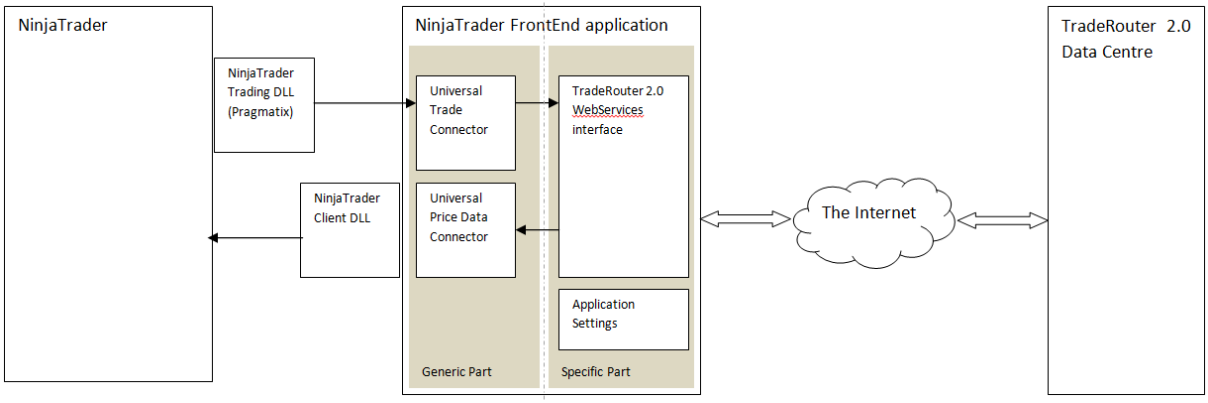
7.2.2 Voorbereiding

Voor de FrontEnd product is het ook van belang om goed voor te bereiden. Ook hier is de insteek om fouten te voorkomen en een duidelijk beeld te creëren over hoe het product er uit moet komen te zien. Deze zaken worden allemaal vastgelegd in een Requirements document.

Aan de afbeelding, in de vorige paragraaf, is te zien dat er nog steeds met het gelaagde architectuur wordt gewerkt. De lagen bestaan nu uit:

- NinjaTrader Client API
- NinjaTrader FrontEnd Application
- NinjaTrader Trading DLL

Via de DLL's is communicatie tussen de lagen mogelijk. Met het TradeRouter ActiveX control is communicatie mogelijk met de TradeRouter. Het is dus de bedoeling dat iedere laag onafhankelijk van elkaar kan werken. Dit wordt voornamelijk gedaan omdat er in de toekomst een vervanger komt van de TradeRouter.

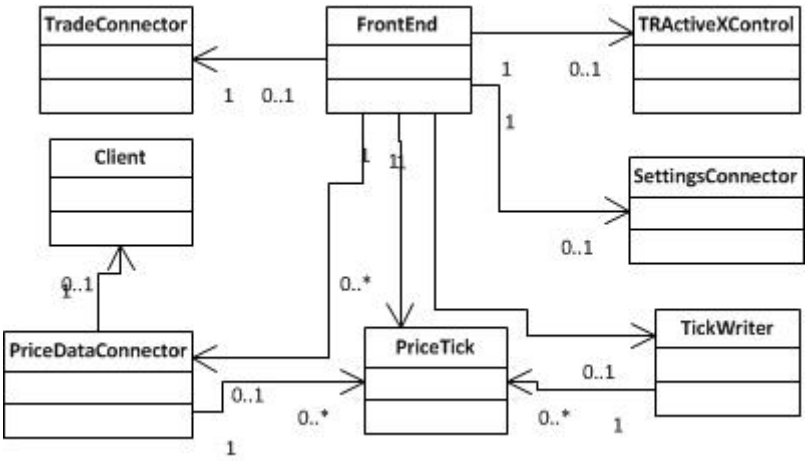


In het Requirements document wordt dieper ingegaan op het ontwerp en de werking ervan.

7.2.3 Realisatie

De realisatie van de FrontEnd applicatie verliep voorspoedig. Er zijn toch wel enkele punten geweest die voor wat vertraging hebben gezorgd, maar dat heeft uiteindelijk weinig impact gehad op het proces.

Het implementeren verliep gemakkelijk omdat niet alleen TradeRouter is geschreven in .Net, maar ook NinjaTrader. Hierdoor was het toevoegen van een DLL aan het project meer dan voldoende. Zo is het gemakkelijk om methodes van beide kanten uit te gebruiken in de FrontEnd.

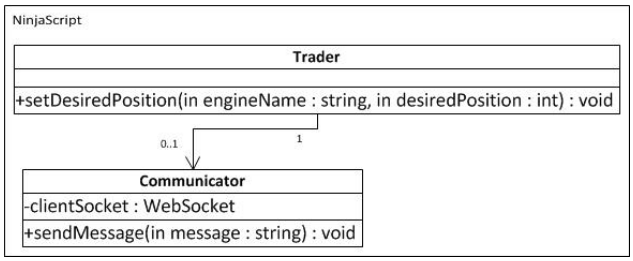


De FrontEnd maakt gebruik van een DLL (Client) van NinjaTrader. Die DLL is geschreven in .Net 3.5, dus zal de FrontEnd ook in .Net 3.5 worden geschreven. Dit zorgt ervoor dat er geen compatibiliteitsproblemen ontstaan.

Verder wordt er gebruik gemaakt van een ActiveX component van TradeRouter. Deze component is geschreven in Visual Basic 6. Het ActiveX component maakt verbinding met TradeRouter om de koersen van TradeRouter op te halen. Verder kan de component ook posities doorsturen. Deze posities bepalen of er financiële producten moeten worden gekocht of verkocht.

De ActiveX component wordt anders geïmporteerd dan de DLL. Het zal aan een Windows Form worden toegevoegd. Door het toevoegen worden de juiste onderdelen aan de applicatie toegevoegd en werkt het naar behoren. De koersdata die binnenkomt via het ActiveX component wordt doorgegeven aan NinjaTrader via de Client DLL van NinjaTrader.

Om de FrontEnd applicatie te kunnen aan spreken is er een DLL ontwikkeld, namelijk de TradeRouterTrading.dll. Deze DLL verstuurt berichten via sockets. Die berichten worden dan opgevangen door de FrontEnd applicatie. Dit gebeurt middels een TCP/IP socket verbinding.



Deze DLL wordt geïmporteerd in een NinjaScript. Met een NinjaScript kan een gebruiker zelf aangeven wanneer er een bericht moet worden verstuurd naar de FrontEnd. De FrontEnd verstuurt het bericht op zijn beurt weer naar TradeRouter.

In het volgende stuk code ziet u hoe de DLL gebruikt wordt in een NinjaScript. In dit voorbeeld wordt er een Trader object aangemaakt. Er worden geen parameters meegeven in de constructor, omdat de FrontEnd applicatie lokaal draait. De host en de port worden dan ingesteld op '127.0.0.1' en '3333'. Dit zijn de standaard waarden. In dit voorbeeld wordt er bij iedere update van de bar(koers), een positie ingenomen. Dit is een heel simpel voorbeeld om de DLL te gebruiken.

```
#region Using declarations
using System;
using System.ComponentModel;
using System.Diagnostics;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Xml.Serialization;
using NinjaTrader.Cbi;
using NinjaTrader.Data;
using NinjaTrader.Gui.Chart;
using NT2TR;
#endregion

// This namespace holds all strategies and is required. Do not change it.
namespace NinjaTrader.Strategy
{
    /// <summary>
    /// Enter the description of your strategy here
    /// </summary>
    [Description("Enter the description of your strategy here")]
    public class TRTest : Strategy
    {
        #region Variables
        // Wizard generated variables
        // User defined variables (add any user defined variables below)
        Trader client;
        #endregion

        /// <summary>
        /// This method is used to configure the strategy and is called once
before any strategy method is called.
        /// </summary>
        protected override void Initialize()
        {
            CalculateOnBarClose = true;
            client = new Trader();
        }

        /// <summary>
        /// Called on each bar update event (incoming tick)
        /// </summary>
        protected override void OnBarUpdate()
        {
             client.setDesiredPosition(Instrument.MasterInstrument.Name, -1);
        }
    }
}
```

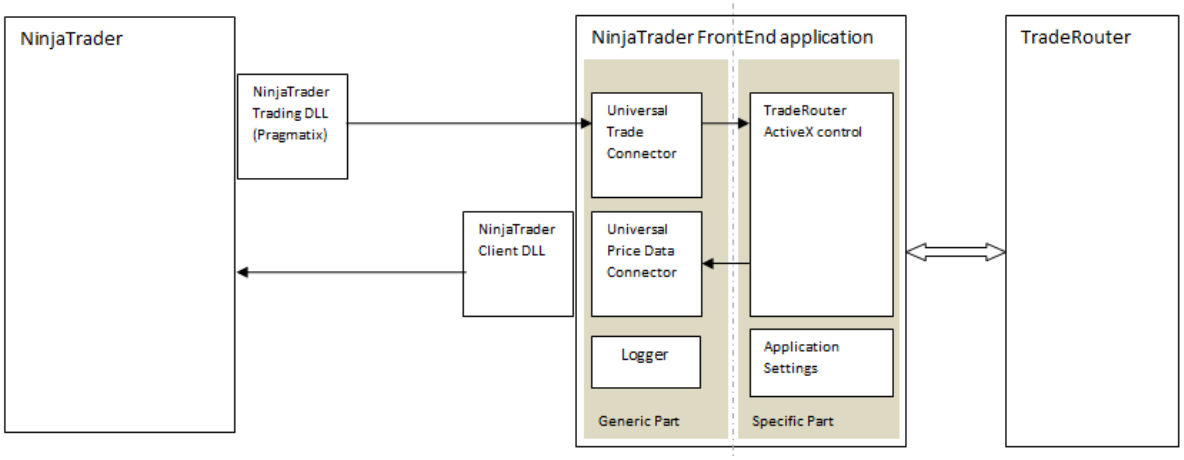
7.2.4 Tegenslagen

Eén van die punten was het versturen van handelssignalen vanuit een NinjaScript. Ik had het zo geprogrammeerd dat het bij iedere prijs update, een signaal zou worden gestuurd. Dit was puur voor het testen. Echter gebeurde er niets. Bij het instellen van een NinjaScript op een koers, heb je ook nog instellingen wanneer het zou moeten gaan werken. Bijv. hoeveel minuten koers informatie er minimaal nodig voordat het script mag gaan werken. Deze stond standaard op 20 minuten, terwijl er nog pas enkele minuten data beschikbaar was. Door deze instelling lager te zetten, werkte het script eerder en werden en er signalen verstuurd.

Er was ook nog een verschil tussen 32-bits Windows en 64-bits. Een bepaalde component van TradeRouter moest in een specifieke folder staan. Maar toch werkte het niet op mijn computer. Ik snapte niet goed hoe dat kwam, aangezien ik de documentatie had gevolgd. De bedrijfsbegeleider keek er even naar en had meteen door dat het in een andere folder moet staan omdat ik op 64-bits Windows werkte.

Een ander punt was het versturen van de koersdata van TradeRouter naar NinjaTrader. Alles verliep goed, bij de eerste koers die ik verstuurde. Ik had NinjaTrader zo ingesteld als de documentatie voorschreef en de koersdata kwam binnen. Toen er meer koersen werden toegevoegd, leek het erop dat NinjaTrader het niet aan kon. Na veel uitproberen, koersen weghalen er weer bij doen, kwam er een patroon in zicht. Koersen met een naam die een getal of een speciaal teken bevatten, die zorgen ervoor dat NinjaTrader het niet aankan. Door de getallen en speciale tekens weg te filteren, werkt NinjaTrader weer soepel. Het maakt niet uit hoeveel koersen het zijn, zolang er maar geen getallen en/of speciale tekens in zitten.

Na deze problemen te hebben ontdekt en verholpen, is de realisatie verder soepel verlopen. Er is ook een class gemaakt voor het loggen van bepaalde gebeurtenissen in de FrontEnd applicatie. Deze class is in de vorm van een Singleton, zodat het overal in de code kan worden gebruikt. Dit was nog een wens van de bedrijfsbegeleider die tijdens de realisatie tot stand is gekomen. De uiteindelijke versie ziet er als volgt uit.



8. Invoering

Het oplossingsplan is gerealiseerd. Dat houdt in dat het nu tijd is om de invoering onder handen te nemen. De FrontEnd dient aan een aantal eisen te voldoen en dat wordt tijdens de invoering uitvoerig gecontroleerd. Als alles goed verlopen is, dan kan er aan de afronding worden gewerkt.

8.1 Testen

Voordat alles ingevoerd kan worden op een bestaand systeem, moet er worden getest. Omdat het om financiële producten gaat, moet er heel streng worden getest. Dit is niet alleen een eis uit Capital Intervest, maar ook die van de AFM. Anders kunnen vele klanten hun geld verliezen door fouten in de software.

Het gebruiken van automatische handelsapplicaties is non-stop. De applicaties draaien dag en nacht, omdat er de hele dag door gehandeld kan worden. In Azië openen de beurzen wanneer het nog nacht is in Nederland en in Amerika zijn de beurzen nog open nadat ze in Europa gesloten zijn. Daarom is het van belang dat de applicatie blijft draaien en excepties kan opvangen en afhandelen.

Veel klanten huren een stukje server in bij NetConnex. NetConnex is een bedrijf waar Capital Intervest veel mee samenwerkt. Ze verhuren stukjes server, waarop een virtuele machine draait. Die virtuele machines draaien allemaal Windows Server 2003. Klanten huren de serverruimte, zodat ze niet hun eigen computer dag en nacht aan hebben staan. Daarom zal er ook een test gedaan worden om de FrontEnd een lange tijd op Windows Server 2003 te draaien.

Uiteraard moet het installeren van de FrontEnd zonder problemen verlopen. Hiervoor wordt een handleiding geschreven. De bedoeling is dat een gebruiker de handleiding volgt, zodat hij de FrontEnd zonder problemen kan gebruiken. Dit is van installatie van de FrontEnd tot het koppelen en gebruik maken van TradeRouter en NinjaTrader.

In de toekomst zou er zonder te veel werk de FrontEnd applicatie gekoppeld kunnen worden aan een ander brokerapplicatie. Dit zou inhouden dat TradeRouter vervangen kan worden. Een TradeRouter 2.0 zou dan bijvoorbeeld ook kunnen koppelen met NinjaTrader. Dit is een test dat na het afronden van het project zou kunnen plaatsvinden, aangezien de scope van het project alleen betrekking heeft tot TradeRouter.

8.2 Resultaat

De FrontEnd moet uiteraard gedistribueerd worden. Dit gebeurt door middel van een setup. Er wordt van de FrontEnd een setup gemaakt, waarbij alle bijbehorende componenten worden toegevoegd. De handleiding voor de FrontEnd en de DLL voor de NinjaScript worden er ook aan toegevoegd. Zo zijn alle vereiste componenten samengevoegd tot één setup.

De setup installeert de FrontEnd en registreert de COM componenten. Verder wordt er een snelkoppeling op het bureaublad en in het start menu gezet. De setup controleert verder of er een oudere versie is en die in dat geval de oude installatie verwijdert.

Voor een succesvolle invoering zal ik collega Ard van der Pijl laten testen. Hij zal samen met mijn handleiding, de FrontEnd werkend krijgen. De handleiding zorgt ervoor dat hij weet hoe hij de installatie moet uitvoeren en hoe hij de FrontEnd uiteindelijk koppelt. Dit zijn de resultaten nadat hij de handleiding heeft doorlopen:

- Volgorde van de handleiding maakt het onduidelijk
- Account dient rechten te hebben voor koppeling

Met deze resultaten moeten er nog een aantal punten worden verbeterd. Dit zal uitgevoerd worden en de test zal herhaald blijven worden totdat er geen fouten en of onduidelijkheden meer zijn geconstateerd. Zo kan er een kwalitatief goed product worden geleverd.

De volgorde van de handleiding wordt aangepast. Het was eerst onduidelijk, omdat deze na de installatie van de FrontEnd meteen overging naar het maken van een NinjaScript. Vervolgens moest je NinjaTrader koppelen met de FrontEnd. Hierdoor ging het van het één naar het ander. Dit werd opgelost door eerst de installatie uit te voeren en de koppeling te maken. Daarna wordt er pas naar een NinjaScript gekeken en het gebruik daarvan.

Een gebruiker van TradeRouter, dient het recht te hebben om een externe applicatie verbinding te laten maken met de TradeRouter. Hier kwamen we achter nadat Ard ingelogd had in de TradeRouter. Hij startte vervolgens de FrontEnd die geen verbinding maakte. Nadat ikzelf in TradeRouter inlogde en de FrontEnd startte, maakte de FrontEnd wél verbinding. Dit had te maken met het feit dat mijn account wel het recht had om externe applicaties toe te laten en het account van Ard niet. Het idee achter deze beperking is om gebruikers expliciet toestemming te geven om hun TradeRouter door eigen software (via het ActiveX control) te besturen. Capital Intervest zal dit recht alleen verstrekken aan gebruikers waarvan zij zeker weet dat zij technisch voldoende onderlegd zijn om trading op een dergelijke manier te automatiseren.

Verder waren er geen punten meer die de applicatie deed vastlopen. De FrontEnd en de handleiding worden aan de begeleider gegeven zodat hij het verder kan beoordelen.

Wanneer de kwaliteit als voldoende wordt beschouwd door de begeleider, dan is de invoering voltooid. Doordat een collega doet alsof hij een klant is die het product in gebruik neemt, kunnen we zeggen dat een klant het kan gebruiken. Omdat de begeleider en de collega het proces beiden hebben doorlopen en hebben besproken met mij, is het goed bekend binnen het bedrijf. Hierdoor is de invoering compleet.

9. Advies

9.1 Conclusies

Om tot een goed werkende koppeling te komen, tussen een brokerapplicatie en technische analysepakket, is grondig onderzoek naar het technische pakket noodzakelijk. Dit komt goed naar voren, omdat er na een onderzoek naar MultiCharts, realisatie mogelijk leek. Echter tijdens het implementeren kwamen er wat fouten naar voren. Na contact te hebben opgenomen met MultiCharts Inc. werd het duidelijk dat de API van MultiCharts waar Capital Intervest mee werkt, te oud is voor de huidige MultiCharts. Als dit eerder aan bod zou zijn gekomen, dan was het eerder duidelijk geweest dat er niet met MultiCharts gewerkt kan worden.

Door NinjaTrader, een ander technische analysepakket, goed te onderzoeken, is het duidelijk dat NinjaTrader een goede keuze is. Dit komt omdat de technologie van NinjaTrader dezelfde is als die van TradeRouter, namelijk .Net. Dit zorgt ervoor dat er met minimale inspanning een goed werkende koppeling kan worden gemaakt.

Dit reflecteert ook nog naar de koppeling tussen MultiCharts en TradeRouter. Hiervoor moest namelijk meer moeite gedaan worden. Er moest een vertaalslag komen tussen Unmanaged C++ en Managed C#. Dit zorgt voor meer werk en mogelijk meer fouten in het Unmanaged gedeelte.

De gelaagde architectuur heeft goed bijgedragen aan het feit dat de FrontEnd onafhankelijk kan werken. NinjaTrader en TradeRouter hoeven niet aan te staan voor de FrontEnd om het te draaien. Dit zorgt er ook voor dat als één van de systemen stopt, dat de anderen gewoon door kunnen gaan. Met het oog op de toekomst, kan de FrontEnd ook ingezet worden voor een vervanger van TradeRouter. Door de gelaagde architectuur kan de FrontEnd met minimale aanpassingen zo ingezet worden op een nieuw platform.

9.2 Aanbevelingen

Een goed leermoment voor Capital Intervest is het compatibiliteitsprobleem van MultiCharts. Wat ze zouden moeten doen is voordat ze aan een project beginnen, eerst contact op te nemen met het bedrijf achter het technische analysepakket. Dit zou er naar toe kunnen leiden om toekomstige knelpunten al vroeg te voorzien.

Nog een klein punt ter verbetering zou zijn om de documentatie voor verschillende Windows versies geschikt te maken. Bijvoorbeeld dat er een verschil in een 32-bits en 64-bits Windows zit of Windows Vista en Windows 7. Daar zou dan eerst ook op getest moeten worden. Zo voorkom je dat je bepaalde hard- en software uitsluit.

Persoonlijke Evaluatie

Voordat ik begon aan de stage wist ik nog niet goed wat ik kon verwachten. Ik vind de beurs interessant, maar echte kennis van beurzen, beleggen enz. was mager bij mij. En wat een bedrijf in de beleggingswereld met software kan doen, was me ook niet duidelijk.

Wat ik wel wist is dat Capital Intervest nog maar 5 jaar bestaat, dus dat het nog een klein bedrijf is. Dit sprak me erg aan, omdat ik uit ervaring weet dat je dan iedereen goed leert kennen in het bedrijf. Je bent dan echt iemand in het bedrijf en niet de zoveelste werknemer. Dit was ook inderdaad zo het geval bij Capital Intervest. Ik voelde me snel thuis en was snel ingewerkt.

In het begin werd mij voorgesteld om het boek “Beleggen voor Dummies” te lezen. Dit bleek uiteindelijk een hele goede tip. Ik leerde in korte tijd wat er allemaal komt kijken in de beleggingswereld. De financiële producten, regels, aanbieders, autoriteiten enz. Zo kreeg ik meer grip op wat er gebeurde en kon ik beter mee praten met mijn collegae.

Vervolgens maakte ik kennis met de TradeRouter, de applicatie van Capital Intervest. Ik leerde wat het allemaal kon en hoe het gebruikt werd. Dit was noodzakelijk omdat ik er uiteindelijk een koppeling voor zou schrijven. Ook leerde ik zo meer de technische kant achter de beurzen. Hoe je bepaalde koersen kunt volgen, waar je de data vandaan kunt halen en hoe je orders kunt plaatsen. Ik begon me een beginnende belegger te voelen.

De volgende stap was het onderzoeken van MultiCharts. Wat ik onderzocht was uit welke technologie de API bestaat en hoe ik deze zou kunnen koppelen aan de TradeRouter. De grootste uitdaging lag in het koppelen van C++ met C#. Al snel kwam ik achter het feit dat er velen manieren waren. Dit motiveerde mij om uit te zoeken wat de beste manier is.

Helaas was ik er tijdens het onderzoek naar MultiCharts niet achter gekomen dat de API waarmee ik werkte niet geschikt is voor de huidige versie van MultiCharts. Ik kwam er pas achter tijdens het implementeren. Dit zorgde ervoor dat ik veel tijd kwijt was aan het zoeken naar de oorzaak. Zo raakte ik enigszins gedemotiveerd. Het leek erop alsof ik alles voor niets had gedaan, maar uiteindelijk zag ik het als een bepaald idee uitsluiten. Ik kon weer opnieuw beginnen met een ander technische analysepakket. Hierdoor raakte ik een beetje van slag, omdat ik bang was dat ik het niet meer op tijd af zou krijgen. Uiteindelijk is het toch goed gekomen en kon ik het hoofd koel houden. Ik heb ervan geleerd dat er tegenslagen zijn in een project en dat het een motivatie moet zijn om voor de tegenslagen een oplossing te vinden. Zo kun je toch doorwerken en het allemaal afkrijgen.

Een hoogtepunt tijdens het afstuderen was de dag dat de AFM langs kwam. Er waren drie inspecteurs langs gekomen. Ze kwamen inspecteren of wij voldoen aan de eisen voor een Broker vergunning. We waren allemaal strak in het pak om zo professioneel mogelijk over te komen. Het was een spannende dag en uiteindelijk hebben we de vergunning gekregen.

Wanneer ik naar de competenties kijk, merk ik dat het één en ander beter is geworden. Samenwerken en professioneel handelen is nooit een probleem geweest. Collegae vragen aan mij of ik iets tussendoor kan doen en dat doe ik zonder problemen. Op mijn beurt vraag ik aan hen of ze mijn applicatie kunnen testen. Dat gebeurd ook zonder enkele tegenspraak

Samenwerken zit er dus goed in. Dat er halverwege het project voor een ander software pakket wordt gekozen en ik daar zonder moeite mee door ga, toont ook aan dat professioneel handelen geen probleem is.

Doordat ik tijdens mijn onderzoek start met analyseren, dan overga op het ontwerpen en tenslotte doorga met het realiseren, is het wel duidelijk dat ik ook goed methodisch handel. Het volgen van deze volgorde zorgt ervoor dat er een duidelijk architectuur voor de software is ontworpen. Ook het tien stappenplan draagt bij aan het goed methodisch handelen.

Ik heb ook het analyseren moeten verbeteren, aangezien ik halverwege het project over moest stappen naar een ander software pakket. Dit vereiste een zeer grondig analyse om zeker te zijn dat ik de goede richting op ging.

Met de resultaten van de onderzoeken kon ik verantwoorden waarom ik voor een bepaald methode heb gekozen. Hiermee adviseer ik dan ook hoe er te werk moet worden gegaan. Dit bewijs ik tevens ook door een ontwerp te maken en het vervolgens te gaan realiseren. Door minder dan 20% af te wijken van het ontwerp, na realisatie, kan ik zeggen dat ik het ontwerpen goed beheers.

Ik had verder mijn eigen applicatie verbeterd en fouten eruit gehaald, totdat het een goed product werd. Het lukte mij om dit binnen het geplande tijd te halen, waardoor te zeggen valt dat het realiseren ook een behaalde competentie is. Verder heb ik tussendoor ook een andere opdracht uitgevoerd. Ik moest een applicatie van iemand anders verbeteren en fouten oplossen. Hiermee bewijs ik dat goed software kan beheren en ook weer goed kan analyseren.

Als ik terugkijk op mijn afstudeerperiode kan ik zeggen dat ik vooral heb geleerd hoe het is om werknemer te zijn van een bedrijf. Dit heb ik hier bij Capital Intervest veel meer ervaren dan bij mijn vorige stage. Dit komt voornamelijk omdat ik wat opdrachten tussendoor kreeg en mee hielp met support. Nu weet ik ook zeker dat ik de competenties goed beheers en dat ik ben klaar om te gaan starten als software engineer.

Literatuurlijst

Boeken

De volgende boeken hebben als input gediend tijdens de uitvoering van dit onderzoek:

- Kanis, Matthuis - *Beleggen voor Dummis* - 2e druk - Pearson Education
- Kempen, Piet & Keizer, Jimme - *Competent afstuderen en stagelopen* – 4e druk – Noordhoff Uitgevers

Achternaam auteur, voornaam - Titel van het boek. Versie van druk. Uitgever.

Internet bronnen

De volgende websites hebben als input gediend tijdens de uitvoering van dit onderzoek:

- Capital Intervest – Geraadpleegd op februari – maart 2012 - http://www.capitalintervest.nl/ - Over Capital Intervest, Begrippenlijst
- Dzone – Geraadpleegd op maart 2012 - http://dzone.com/snippets/calling-unmanaged-c-function-c
- Dream in code – Geraadpleegd op maart 2012 - http://www.dreamincode.net/forums/topic/38890-activex-with-c%23/
- Microsoft Developers Network – Geraadpleegd op maart 2012 - http://blogs.msdn.com/b/borisj/archive/2006/09/28/769708.aspx
- MultiCharts – Geraadpleegd op maart 2012 - http://www.multicharts.com/ - Support, OwnData
- MetaTrader – Geraadpleegd op april 2012 - http://www.metatrader5.com/en - For Brokers
- NinjaTrader – Geraadpleegd op april 2012 - http://www.ninjatrader.com/ - Custom Development, Forum

Naam van website. Geraadpleegd op dag maand jaar, adres website. Pagina(s).

Bijlagen

I - PID MultiCharts

II - RoadMap CI

III - Unmanaged naar Managed

IV - Requirements data adapter

V - Een alternatieve technische analysepakket

VI - Requirements FrontEnd

VII - Handleiding voor FrontEnd koppeling

VIII - Nevenactiviteiten

Bijlage I

PID MultiCharts

Project Initiation Document

Unmanaged naar Managed

9-3-2012

Documentinformatie

Auteur	Michael Rademaker
Studentnr	2111026
Datum	07/03/2012
Versie	0.3

Opleiding		Organisatie	
Onderwijsinstelling	Fontys Hogenscholen ICT	Naam bedrijf	Capital Intervest
Opleidingsrichting	HBO ICT Software Engineering		
Adres	Rachelsmolen 1 R1 5612 MA Eindhoven	Adres	Bredaseweg 234 5038 NM Tilburg
Telefoonnr.	0877 877 877	Telefoonnr.	013 571 60 51
Begeleiding	Dhr. M. Franssen	Begeleiding	Dhr. F. Hoogland

Documenthistorie

Revisies

Versie	Status	Datum	Wijzigingen
0.1	Concept	17-02-2012	Opzet eerste versie.
0.2	Concept	02-03-2012	Opzet tweede versie
0.3	Concept	07-03-2012	Opzet derde versie
1.0	Defenitief	09-03-2012	Feedback van begeleiders verwerkt

Goedkeuring

Dit document behoeft de volgende goedkeuringen:

Versie	Datum goedkeuring	Naam	Functie	Paraaf
1.0	09-03-2012	Frank Hoogland	Projectbegeleider	
1.0	09-03-2012	Michael Franssen	Docentbegeleider	

Distributie

Dit document is verstuurd aan:

Versie	Datum verzending	Naam	Functie
0.1	17-02-2012	Frank Hoogland	Projectbegeleider
0.1	17-02-2012	Michael Franssen	Docentbegeleider
0.2	02-03-2012	Michael Franssen	Docentbegeleider
0.3	07-03-2012	Frank Hoogland	Projectbegeleider
0.3	07-03-2012	Michael Franssen	Docentbegeleider

Managementsamenvatting

Doel van dit document

Dit document heeft tot doel het project te definiëren en de beoordeling van het succes van het project mogelijk te maken.

De twee belangrijkste punten van dit document zijn:

- om er zeker van te zijn dat het project een gezonde basis heeft voordat Michael Rademaker (verder te noemen als de afstudeerder), gevraagd wordt zich aan het project committeren;

- om te dienen als basisdocument op grond waarvan de afstudeerder, Frank Hoogland (verder te noemen als Projectbegeleider) en Michael Franssen (verder te noemen als de Docentbegeleider) de voortgang en wijzigingen kunnen toetsen en bewaken en vragen betreffende geldigheid van het project tijdens de uitvoering ervan kunnen beoordelen.

Aanleiding

Er komen steeds meer beleggers op de beleggingsmarkt die in de vele financiële producten willen handelen. Capital Intervest, een bedrijf dat software ontwikkeld om automatisch te handelen, heeft als doel om de toenemende groep beleggers te helpen bij het beleggen. Dit doen ze door software oplossingen te bedenken.

Dit project op het gebied van ophalen en versturen van koersdata is tot stand gekomen, omdat tegenwoordig niet alleen de beurs de plaats is waar er gehandeld wordt; er komen steeds meer instanties die een eigen ‘beurs’ hebben waarop gehandeld wordt. Zo’n instantie wordt een broker genoemd.

Tevens versturen de meeste brokers data via een eigen protocol. Hierdoor zijn er verschillende protocollen en wordt het lastig om deze te gebruiken in een technische analysepakket, in dit geval MultiCharts, die handelsadviezen generen. Een pakket als MultiCharts ondersteunt namelijk slechts koppelingen met algemene dataleveranciers, plus enkele grotere brokers.

Als dit project is uitgevoerd is niet alleen een koppeling tussen een specifieke broker en een MultiCharts gerealiseerd, maar zal het toekomstig ook eenvoudiger worden om koppelingen tussen MultiCharts en andere brokers op te zetten.

Globale aanpak

Er wordt aan de hand van het Tienstappenplan gewerkt. Voor het beginnen aan het project, zijn heeft er al een externe oriëntatie en een intake gesprek plaatsgevonden. Zo is er immers aan het project begonnen. In het verloop van het project zullen de andere stappen terug komen.

In dit project wordt er onderzoek gedaan naar de huidige situatie van Capital Intervest en naar de mogelijkheden van een universele data adapter voor nu en in de toekomst. Er wordt onderzocht hoe de gebruikte applicatie in elkaar zit en werkt.

Met die bevindingen wordt er verder onderzocht wat moet gebeuren en wat nodig is om een universele data-adapter te bouwen. Er moet rekening mee worden gehouden dat de API met ActiveX in C++ werkt, technologie wat enigszins verouderd is.

Daarnaast is het van belang dat de universele data adapter met een modern programmeer framework wordt gebouwd. Hiervoor is er gekozen voor .Net 4.0 Framework, omdat .Net ook ActiveX elementen bevat. Dit is vooral een wens om in de toekomst gemakkelijk aanpassingen/uitbreidingen te kunnen maken.

Tenslotte is het zeer van belang dat er uitvoerig wordt getest, aangezien foute handelsadviezen zullen leiden tot mindere beleggingsresultaten. Het testen wordt gerealiseerd door data op te halen en door te sluizen via de data adapter.

Globale kosten en doorlooptijd

Er zijn bij dit project geen hoge kosten van toepassing dus het project brengt wat dat betreft ook geen risico's met zich mee. De doorlooptijd van het project is 17 weken, met daarin 85 werkdagen. De deadline is 8 juni 2012.

Inhoudsopgave

1. Inleiding

1.1 Doel van dit document

Dit document is opgesteld om alle relevante informatie en uitgangspunten van het project vast te leggen. Het heeft tot doel het project te definiëren en de beoordeling van het succes van het project mogelijk te maken.

Dit Projectinitiatiedocument (of PID) behandelt de volgende fundamentele aspecten van het project:

- Wat beoogt men met het project te bereiken?
- Waarom is het belangrijk om deze doelstellingen te bereiken?
- Wie zijn er betrokken bij het managen van het project en wat zijn hun rollen en verantwoordelijkheden?
- Hoe en wanneer zullen de maatregelen die in dit PID besproken worden gerealiseerd worden?

Het document wordt gebruikt:

- om er zeker van te zijn dat het project een gezonde basis heeft voordat de Stuurgroep gevraagd wordt zich aan het project committeren;
- om te dienen als basisdocument op grond waarvan de afstudeerder en de Projectmanager de voortgang en wijzigingen kunnen toetsen en bewaken en vragen betreffende geldigheid van het project tijdens de uitvoering ervan kunnen beoordelen.

1.2 Opbouw van dit document

Bepaalde onderdelen van dit document iteratief zijn en dus nieuwe versies zullen krijgen tijdens de voortgang van het project. Andere onderdelen zullen niet meer veranderen, vandaar dat het Projectinitiatiedocument verdeeld is in twee secties: een statisch gedeelte en een dynamisch gedeelte:

Het "statische" deel bestaat uit de hoofdstukken en bijlagen:

- Achtergrond (Hoofdstuk 2)
- Projectdefinitie (Hoofdstuk 3)
- Projectorganisatiestructuur (Hoofdstuk 4)
- Communicatieplan (Bijlage A)
- Planning (Bijlage B)

Het "dynamische" deel bestaat uit de bijlagen:

- Initiële Business Case (Bijlage C)
- Initieel Projectplan (Bijlage D)

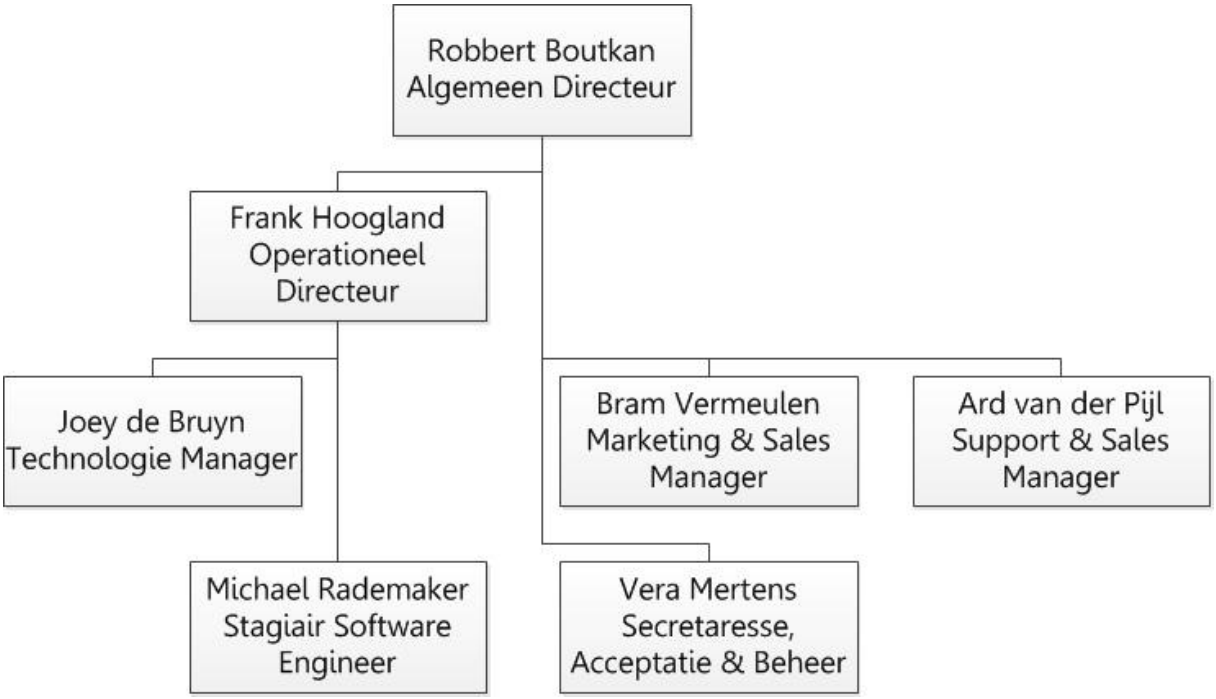
2. Achtergrond

2.1 Capital Intervest

Capital Intervest B.V. is een onafhankelijke aanbieder van producten en diensten voor de wereldwijde beleggersmarkt. Capital Intervest B.V. onderscheidt zich door een ongekende range van financiële markten en producten beschikbaar te stellen voor iedere belegger. Dit door middel van software dat exact wordt afgestemd op de wensen van de klant. Het beschikbaar stellen van die range is voor zowel de professionele belegger als de particuliere belegger.

Capital Intervest heeft een applicatie uitgebracht genaamd TradeRouter. Met TradeRouter kan de belegger, door middel van bepaalde instellingen, geautomatiseerd handelen. Dit zorgt ervoor dat de belegger niet meer achter de schermen hoeft blijven zitten en dat de computer voor hem handelt.

Capital Intervest bestaat uit 6 werknemers. Namelijk een Algemeen Directeur, een Operationeel Directeur, een Technologie Manager, een Support & Sales manager, een Marketing & Sales manager en een Secretaresse die ook voor Acceptatie & Beheer zorgt. Dan komt de afstudeerder als software engineer erbij(zie organigram).



2.2 Project aanleiding

Er wordt gestreefd naar het aanbieden van zoveel mogelijk koersdata aan haar klanten dit is het ideale beeld van Capital Intervest. Zoals besproken is het doel dus om alle brokers op de wereld beschikbaar te stellen aan klanten van Capital Intervest. Zo heeft de klant beschikking tot alle koersdata van de hele wereld en is er niets onbereikbaar.

Om de concurrentie aan te gaan, werkt Capital Intervest het liefst met moderne programmeer technieken. Ze zijn gespecialiseerd in het .Net Framework van Microsoft, waar ze het liefst mee werken.

Dit project op het gebied van ophalen en versturen van koersdata is tot stand gekomen, omdat tegenwoordig niet alleen de beurs de plaats is waar er gehandeld wordt; er komen steeds meer instanties die een eigen 'beurs' hebben waarop gehandeld wordt. Zo'n instantie wordt een broker genoemd.

Tevens versturen de meeste brokers data via een eigen protocol. Hierdoor zijn er verschillende protocollen en wordt het lastig om deze te gebruiken in een technische analysepakket, in dit geval MultiCharts, die handelsadviezen generen. Een pakket als MultiCharts ondersteunt namelijk slechts koppelingen met algemene dataleveranciers, plus enkele grotere brokers.

MultiCharts analyseert meerdere koersgrafieken en genereerd zo handelsadviezen. Om ook MultiCharts te kunnen aanbieden, moet er ook handelsadviezen uit de brokers', waar Capital Intervest mee samenwerkt, koersen gehaald kunnen worden. Deze koersen moeten dus gebruikt kunnen worden in MultiCharts. Het wordt dus een data-adapter die specifiek is voor MultiCharts. Met deze data-adapter kunnen brokers applicaties worden gekoppeld met MultiCharts.

Hiervoor is er een tussenlaag nodig tussen de aangeleverde koersdata en MultiCharts. Dit wordt de universele data-adapter. In het volgende hoofdstuk wordt de data-adapter verder uitgelegd.

3. Projectdefinitie

3.1 Projectdoelstellingen

Capital Intervest heeft het op dit moment erg druk met diverse opdrachten. Hierdoor ontstaat er een tekort aan tijd om onderzoek uit te voeren naar een universele data-adapter. Zoals eerder vermeld, wordt er gestreefd naar een data-adapter die in staat is zonder moeite koersdata binnen te halen van een broker en die door te sluizen naar MultiCharts.

Capital Intervest is een bedrijf dat alle beschikbare koersdata wilt aanbieden aan haar klanten. Met al die koersdata kunnen klanten meer en betere beslissingen nemen met betrekking tot het handelen op de beleggingsmarkt. Hierdoor kan Capital Intervest zich onderscheiden door de aanbieder te zijn met alle koersdata in een overzicht voor de klant.

Wanneer de adapter af is, kan Capital Intervest haar klanten van meer behoeften voorzien. Potentiele klanten die voorheen niet de juiste data bij Capital Intervest konden krijgen, zullen alsnog voorzien kunnen worden.

De voordelen die het project kan bieden zijn:

- Een optimale leeromgeving voor de afstudeerder: De afstudeerder zal meer werk ervaring krijgen en hierdoor beter voorbereid zijn voor de arbeidsmarkt.
- Onafhankelijk zijn van de protocollen die de brokers bieden: Capital Intervest zal moeiteloos alle verschillende formaten van koersdata kunnen gebruiken in haar eigen software.
- Meer koersdata aanbieden aan klanten: Klanten kunnen zo meer en betere beslissingen nemen op de beleggingsmarkt.
- Marktpositie verbeteren: Capital Intervest heeft meer te bieden aan beleggers dan andere partijen, waardoor beleggers bij Capital Intervest terecht komen.
- Kennis vergaren over het koppelen van het .Net Framework met oudere software zoals ActiveX componenten: Zowel voor de afstudeerder als voor Capital Intervest is dit interessant.

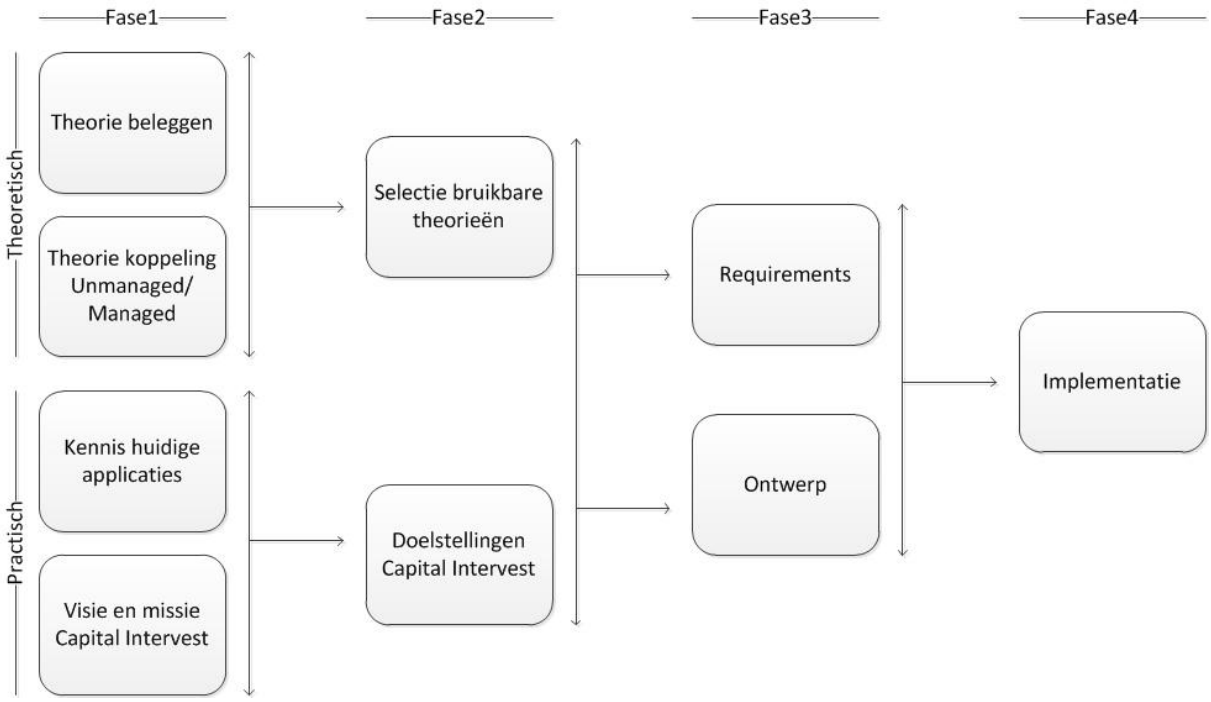
De mogelijkheden die het project kan creëren zijn:

- Nieuwe projecten op basis van de universele data-adapter: Er kunnen nieuwe projecten bedacht worden die gebruik maken van de data-adapter.
- Het verkopen van de universele data-adapter aan bedrijven: De concurrentie of bedrijven waar intensief mee samen wordt gewerkt kunnen interesse hebben in de data-adapter.
- Ideeën die vanuit de afstudeerder komen voor Capital Intervest: Tijdens het ontwikkelen leert de afstudeerder Capital Intervest beter kennen en kan hij ideeën overleggen met Capital Intervest.
- Mogelijke baan bij Capital Intervest voor de afstudeerder: Als het project goed verloopt en Capital Intervest tevreden is over de werkzaamheden, is het zeer waarschijnlijk dat Capital Intervest de afstudeerder een baan aanbiedt.

3.2 Gekozen oplossing of aanpak

In dit project wordt er onderzoek gedaan naar de huidige situatie van Capital Intervest en naar de mogelijkheden van een universele data-adapter. De onderzoeksvraag luid: "Is het haalbaar is om een efficiënte, betrouwbare en universele data-adapter te bouwen om data aan te leveren aan MultiCharts?". De vraag wordt positief beantwoord als na het bouwen van de adapter het mogelijk is om met beperkte inspanning de koppeling te maken tussen MultiCharts en TradeRouter.

Het onderzoek is verdeeld in vier fases die hieronder aan bod komen.



In **Fase 1** wordt er onderzocht waar Capital Intervest nu staat en naar toe wil, op gebiedt van technologie. Daarnaast wordt er onderzocht hoe MultiCharts in elkaar zit. Dit zal in eerste instantie gebeuren door MultiCharts te gebruiken en in te zien hoe er mee gewerkt wordt. Verder wordt er gekeken naar API calls van MultiCharts en hoe deze werken. De API calls werken met ActiveX componenten, dus er zal ook moeten worden gekeken naar ActiveX. Er is gekozen voor het .Net 4.0 Framework in combinatie met C# voor het programmeer framework, omdat er ActiveX elementen in het .Net framework zitten. Hier komt het analyse

Er wordt voornamelijk gekeken of het programmeer framework kan voldoen aan de eisen van de API calls van MultiCharts. Daarnaast moet er nog onderzocht worden hoe de verschillende data formaten kunnen worden opgevangen en omgezet naar één formaat, die dan weer doorgestuurd wordt naar MultiCharts.

Hier komt het analyse aspect van het tienstappen plan in terug.

Daaropvolgend komt **Fase 2** waarin de uitkomsten van Fase 1 bestudeerd worden. In deze fase moet er een goed beeld gecreëerd worden over welke weg het beste ingeslagen kan worden ten aanzien van het gebruik .Net 4.0 Framework met ActiveX. Hierbij moet men denken aan benodigde kennis en kennis die er al is, toepasbaarheid en naar toekomstige

opdrachten. De resultaten van Fase 1 en Fase 2 komen allemaal terug in een conclusie rapport.

Dit is de werkplanning en projectorganisatie. Ook het diepteonderzoek komt erbij kijken.

Die conclusies en keuzes worden opgenomen in **Fase 3**. In deze fase wordt er ook gekeken naar welke architectuur(en) aansluit bij de gevonden oplossing. Uiteindelijk wordt er een Requirements document gemaakt. Daarin staat een ontwerp en een aantal eisen en specificaties over de universele data-adapter, waarin wordt vastgelegd wat en hoe het gaat gebeuren. Het Requirements document moet worden goedgekeurd door de opdrachtgever om verder te mogen met de laatste fase.

Hier uit valt te leiden dat het om het oplossingsplan gaat.

Afsluitend volgt **Fase 4**, waar de resultaten van Fase 3 worden gerealiseerd in een product. Tijdens deze fase is het van belang om eventuele foute beslissingen van Fase 3 snel te ontdekken en het Requirements document aan te passen. Vaak zijn dit fouten die pas tijdens realisatie ontdekt worden. Door hier vroeg bij te zijn zal er minder tijd nodig zijn voor het grondig testen van het product. Dit bespaart tijd en zorgt ervoor dat er meer tijd overblijft voor 'fine-tuning'.

De invoering en afronding komen hierin in terug.

De universele data-adapter kan Capital Intervest meer kennis geven over het .Net 4.0 Framework, evenals over ActiveX componenten. Ook kan er kennis worden vergaard over het toepassen van dit framework naast technische analysepakketten. Tevens zou het een applicatie kunnen zijn die nieuwe opdrachten mogelijk maakt, omdat die opdrachten afhankelijk zijn van de universele data-adapter. De universele data-adapter zou ook puur een advies naar Capital Intervest kunnen zijn bestaande uit feiten en conclusies maar wel gepresenteerd in een applicatie over het .Net 4.0 Framework.

Om een universele data-adapter te bouwen, die ook echt universeel is, zijn de volgende vragen van belang:

- Hoe is MultiCharts opgebouwd?
- Wat zijn de eisen van de API van MultiCharts?
- Hoe kan er vanuit C# (.Net 4.0 Framework) worden gecommuniceerd met ActiveX?
- Wat wil Capital Intervest bereiken met de universele data-adapter?
- Zijn er andere projecten die kunnen ontstaan door de data-adapter?
- Wordt het een applicatie alleen voor intern gebruik, of zullen klanten er ook gebruik van gaan maken?

3.3 Deelvragen onderzoek

Voor het onderzoek zijn diverse deelvragen opgesteld; deze vindt u hieronder.

Deelvragen fase 1-3: Analyse en selectie

- Wat is de huidige situatie van Capital Intervest?
  - Welk programmeer framework worden op dit moment gebruikt binnen Capital Intervest?

  - Wat voor klanten heeft Capital Intervest op dit moment?
  - Wat voor soort opdrachten realiseert Capital Intervest op dit moment?
  - Wat is het huidige doel van Capital Intervest tot het gebruik van nieuwe het .Net 4.0 Framework met oudere technieken zoals ActiveX?
- Wat is ActiveX?
  - Welke framework(en) vallen onder de term ActiveX?
  - Welke voor en nadelen heeft ActiveX?
  - Hoe ziet de ontwikkeling van ActiveX eruit?
  - Hoe kan ActiveX in het .Net 4.0 Framework gebruikt worden?
- Wat is de kennis van Capital Intervest m.b.t. het .Net 4.0 Framework?
  - Welke lopende projecten zijn er?
  - Zijn er al projecten die de koppeling tussen ActiveX en het .Net 4.0 Framework toepassen?
  - Zijn er opdrachten die meerwaarde krijgen door de kennis die vergaard wordt over de koppeling?

Middelen

De middelen die gebruikt worden tijdens het onderzoek zijn:

- Een voorbeeld project
- Deskresearch
- Interviews met de Projectmanager

Bronnen

Boeken:

- Kanis, M. - Beleggen voor Dummies / druk 2

Documenten

- MultiCharts Inc. - Owndata API

3.4 Scope van het project

De volgende partijen zijn zullen baad hebben bij het verwezenlijken van het project:

- Capital Intervest: Er zal meer kennis gewonnen worden over het koppelen van oudere programmeer technieken met moderne technieken. Tevens zal Capital Intervest zijn positie verbeteren qua het totale aanbod van koersdata.
- Klanten: Er is meer koersdata beschikbaar voor MultiCharts waardoor klanten meer handelsstrategieën kunnen ontvangen.
- Brokers (en andere leveranciers van koersdata): Hun koersdata wordt beschikbaar voor MultiCharts, waarmee ze handelsadviezen kunnen genereren.

De volgende partijen zijn betrokken bij het project:

- De afstudeerder: Hij zal het project uitvoeren en de resultaten terug koppelen aan de andere partijen.
- De Projectmanager: Resources beschikbaar stellen voor de afstudeerder en hem begeleiden in het project. Tevens zal hij het project beoordelen.

- De Docentbegeleider: Begeleiden van de afstudeerder door documenten te beoordelen en feedback te geven. Ook zal hij het project gaan beoordelen.

Dit project beperkt zich tot het maken van een universele data-adapter voor MultiCharts. Andere technische analysepakketten komen dus niet aan bod. Met de resultaten van de universele data-adapter wordt er gekeken of het mogelijk is om het .Net 4.0 Framework te koppelen aan ActiveX.

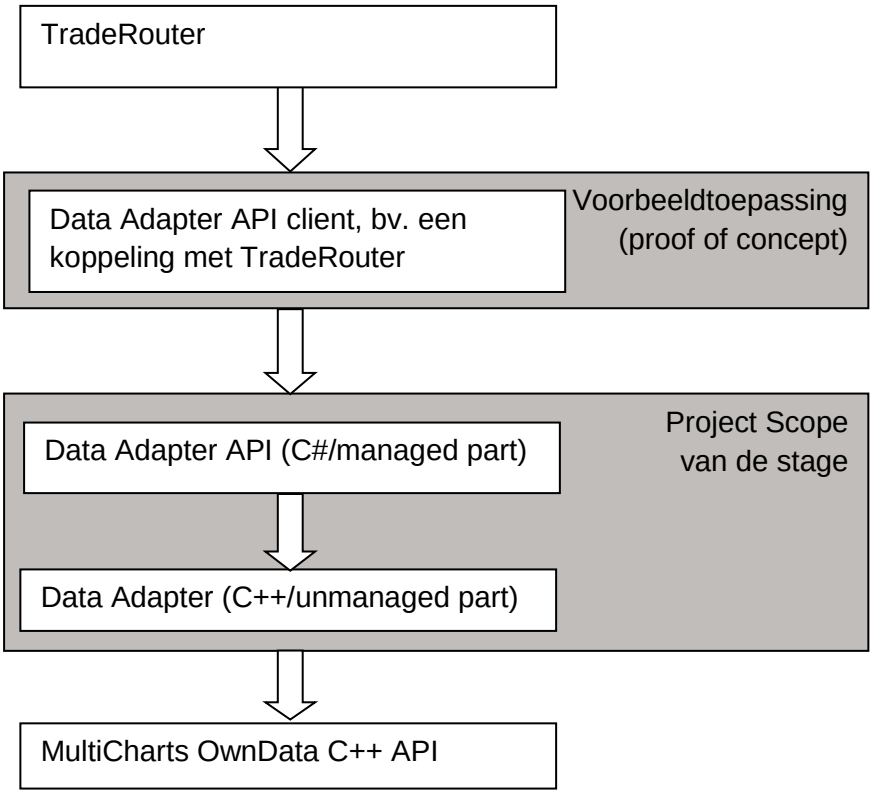
3.5 Eindresultaat

De producten die opgeleverd moeten worden hebben verschillende doelen. Om te beginnen moet er een document komen met daarin de analyses van Capital Intervest, MultiCharts en .Net 4.0 framework. Hierin wordt ook gekeken naar de mogelijkheden van het koppelen van ActiveX met het .Net 4.0 Framework.

Met behulp van de analyse document wordt er een document gemaakt met daarin in conclusies en beargumenteerde beslissingen. Dit wordt samen met de Projectbegeleider doorgenomen en uiteindelijk door hem goedgekeurd.

Verder moet er een document komen, dat de vorm aan neemt van een Requirements document. Dit product vormt een beeld over hoe de applicatie moet gaan werken en hoe het uit gaat zien. De resultaten van het rapport zullen verwerkt worden in een Proof of Concept.

De Proof of Concept een koppeling tussen de Data Adapter API en een broker, in dit geval TradeRouter. Dit samen vormt een implementatie van de universele data-adapter. Voor een andere broker, zou alleen de koppeling geschreven moeten worden en dan tussen de broker en de Data-adapter API moeten staan.



De applicatie kan nieuwe opdrachten tot stand brengen, maar kan ook een puur technisch element zijn dat later door Capital Intervest binnen een eigen applicatie gebruikt kan worden. Daarnaast zou de universele data/adapter ook puur een leerzame ontwikkeling richting Capital Intervest kunnen zijn met betrekking tot het koppelen van het .Net 4.0 Framework met oudere technieken zoals ActiveX.

Verder kan de adapter kan gebruikt worden om aan klanten te laten zien dat het mogelijk is om koersdata van meerdere brokers te kunnen gebruiken in MultiCharts. Hierdoor kan Capital Intervest zich positioneren als een bedrijf met onuitputtelijke informatie.

Uiteindelijk zijn er dus de volgende producten:

- Een analyse-document met daarin omschreven waar Capital Intervest naar toe wilt en mogelijkheden voor een koppeling tussen het .Net 4.0 framework en ActiveX, in het bijzonder de API calls van MultiCharts.
- Een document met daarin conclusies en beargumenteerde keuzes.
- Een Requirements document met daarin een ontwerp van de universele data-adapter, eisen en specificaties.
- Een Proof of Concept dat bestaat uit de universele data-adapter.

3.6 Uitsluitingen

Voor het project zal er alleen een koppeling gemaakt worden tussen Capital Intervest haar eigen TradeRouter en MultiCharts. Andere koppelingen zijn dus uitgesloten.

3.7 Beperkingen

Beschikbaar Budget	Beschikbare resources	Gewenste opleverdatum	Beschikbare Doorlooptijd
600 uur	De IT afdeling van Capital Intervest.	08-06-2012	17 weken

3.8 Afhankelijkheden

De afhankelijkheden van dit project zijn beperkt omdat het onderzoek zelfstandig gedaan kan worden door de afstudeerder. Er is verder geen specifieke software of hardware voor nodig. Tevens hebben de collega's altijd wel tijd beschikbaar en daardoor kan de afstudeerder tijdig aan de bel trekken om door te kunnen werken.

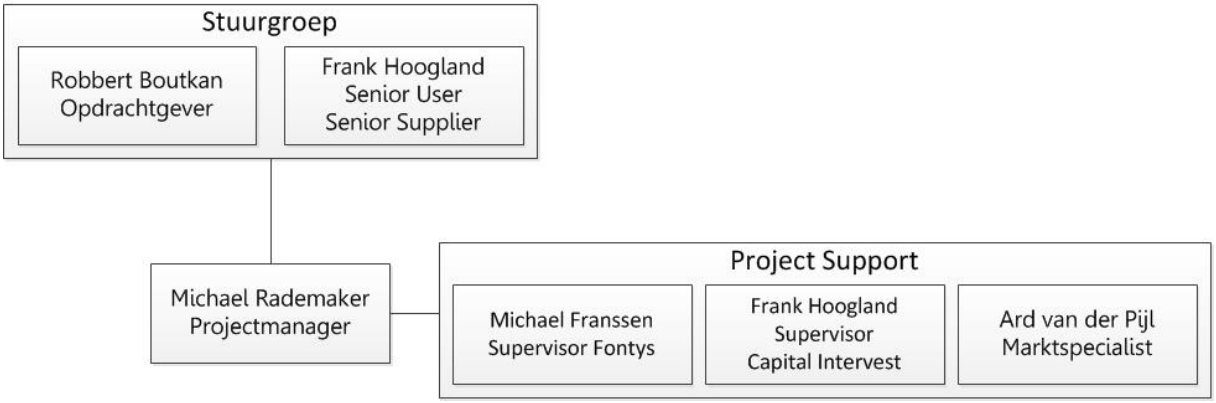
3.9 Randvoorwaarden

- Capaciteit voor het kunnen opvangen van verschillende protocollen
- Kennis van C++ beheren (ActiveX)

3.10 Aannames

De enige aanname is dat er voldoende tijd beschikbaar gemaakt kan worden om het volledige onderzoek en de nodige uitwerkingen te verrichten.

4. Projectorganisatiestructuur



4.1 Projectrollen

Rol	Project gerelateerde taken	Rolbeschrijving
Opdrachtgever Robbert Boutkan	Het beantwoorden van vragen die in relatie zijn met het project. Het goedkeuren van het eindresultaat	De Opdrachtgever, Capital Intervest met als vertegenwoordiger Robbert Boutkan, geeft een mondelinge briefing van de opdracht. De opdrachtgever is de verzoekende partij welke duidelijk zijn wensen en eisen moet overbrengen.
Senior User Frank Hoogland	Opdrachten geven om in Fases het project te verwezenlijken	De Senior User geeft leiding aan de gebruikers die met de resultaten van het project moeten werken. Verder vertaald hij de wensen en eisen van de stuurgroep naar een technische opdracht omschrijving die aan de projectmanager wordt overhandigd.
Senior Supplier	De resources beschikbaar maken om het project te kunnen uitvoeren.	De Senior Supplier is verantwoordelijk voor het vrijgeven van de resources benodigd om het project uit te voeren.
Projectmanager Michael Rademaker	Afspraken maken over doelstellingen en randvoorwaarden. Beheersen van de voortgang van het project.	De projectmanager, in dit geval Michael Rademaker, is degene die de voor sturing binnen de projectgroep zorgt. Tevens is hij het aanspreekpunt voor de projectgroep.
Marktspecialist Ard van der Pijl	Projectgroep ondersteunen met informatie over de beleggingsmarkt	De Markt specialist behoort bij de support afdeling. Hij kent de producten die binnen Capital Intervest gebruikt worden zeer goed, omdat hij ze zelf gebruikt en ook klanten op weg helpt met de applicaties. Verder heeft hij zeer veel kennis over de beleggingsmarkt, de markt waar Capital Intervest zich in bevindt.
Supervisor Capital Intervest Frank Hoogland	Toezichthouden op het proces. Toezicht te houden op de kwaliteit van de eindproducten.	De Supervisor van Capital Intervest biedt eventueel ondersteuning aan de projectmanager, controleert de voortgang van het totale project.
Supervisor Fontys Michael Franssen	Toezichthouden op het proces. Ervoor zorgen dat de eisen van Fontys terug komen in het project.	De Supervisor van Fontys biedt eventueel ondersteuning aan de projectmanager, controleert de voortgang van het totale project. Tevens houdt hij toezicht of er aan de eisen van Fontys wordt voldaan.

4.2 Contactgegevens

Om er zeker van te zijn dat alle partijen met elkaar contact kunnen opnemen, zijn hieronder de contactgegevens te vinden:

4.2.1 Gegevens stagiair

Michael Rademaker
Korte Nieuwstraat 58
5014HA Tilburg
06 27 07 37 29
michael@pragmatix.nl

4.2.2 Organisatiegegevens

Capital Intervest B.V.
Bredaseweg 234
5038NM Tilburg
013 571 60 51

Contactpersoon
Frank Hoogland
013 711 51 04
frank@capitalintervest.com

4.2.3 Opleiding gegevens

Fontys Hogescholen ICT
Rachelsmolen 1
5612MA Eindhoven R1
0877 877 877

Contactpersoon
Michael Franssen
088 508 93 13
michael.franssen@fontys.nl

Bijlage A: Communicatieplan

Inleiding

In dit communicatieplan wordt vastgelegd op welke manieren er met elkaar wordt gecommuniceerd. Ook wordt er vastgelegd wie er met elkaar communiceren. Dit gaat over alle betrokken partijen bij het project en gaat ook puur over communicatie in belang van het project.

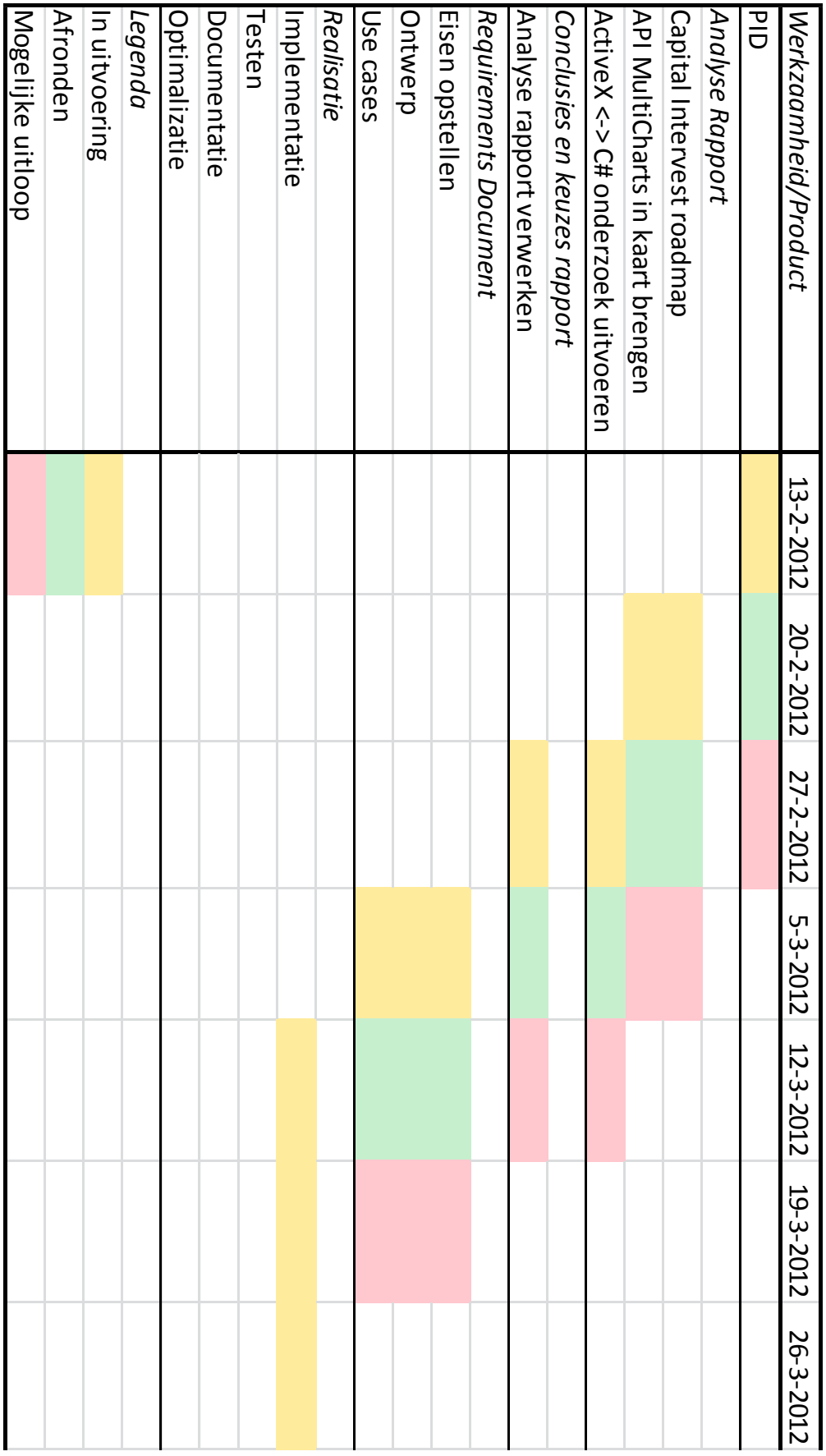
Belanghebbenden bij het project

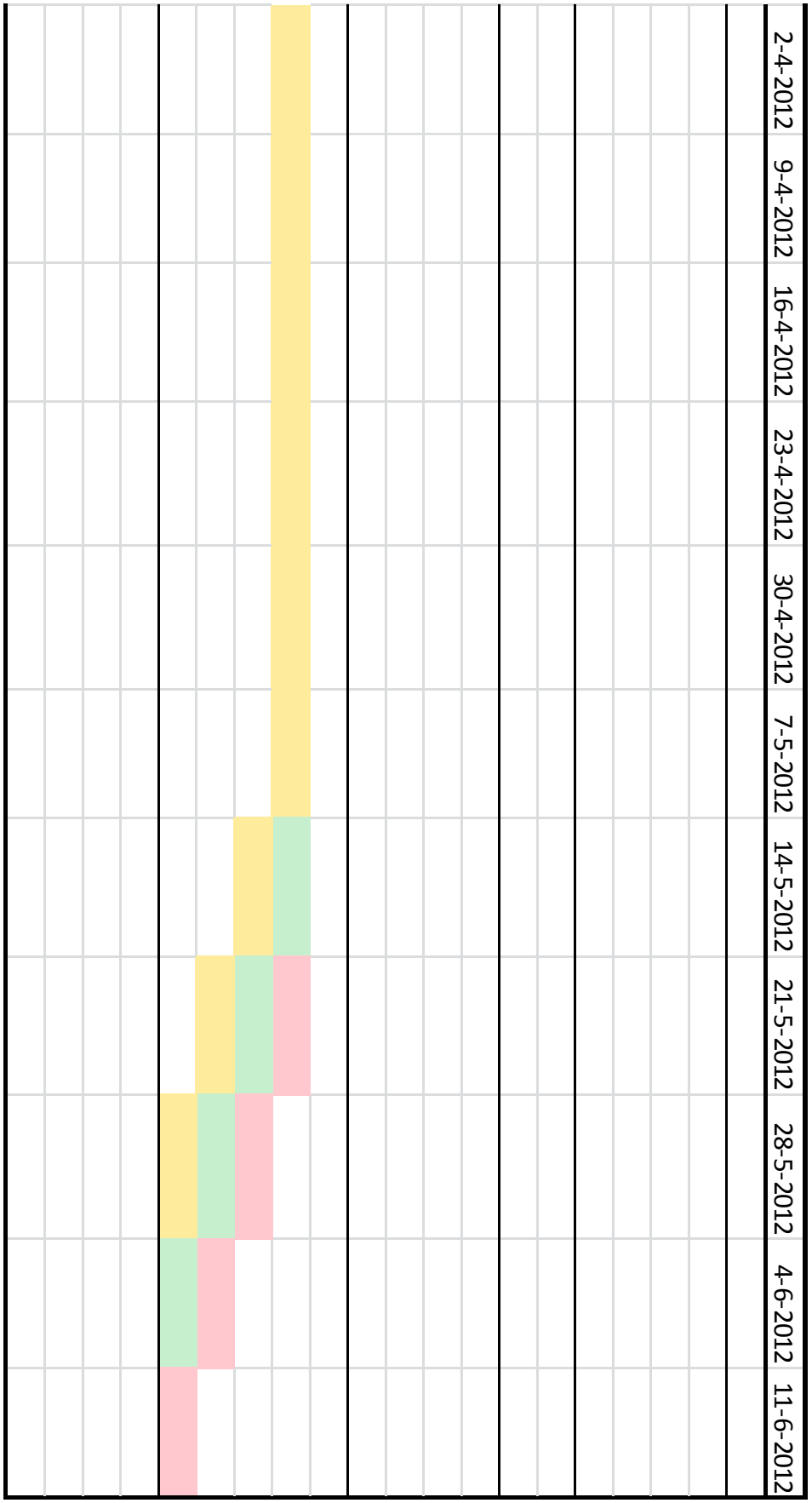
Wie	Namens	Belang	Communicatievorm
persoon	*groep, afdeling*	*Welk belang bij project?*	*Hoe wordt deze persoon betrokken bij het project?*
Marktspecialist	Projectmanager	Advies	Mondeling
Projectmanager	Stuurgroep	Rapporteren	Email, mondeling
Supervisor Capital Intervest	Projectmanager	Adviseren/rapporteren	Email, mondeling, bellen
Supervisor Fontys	Projectmanager	Adviseren/rapporteren	Email, mondeling

Communicatiekanalen

Van	Naar	Informatie	Medium	Frequentie of data
Persoon of groep	*Persoon of groep*	*Soort informatie*	*Email, telefoon, memo, rapport*	
Stuurgroep	Projectmanager	Opdrachtgeving	Email, mondeling	Maandelijks
Projectmanager	Supervisor Capital Intervest	Rapportage	Email, bellen	Wekelijks/ dagelijks
Projectwaarnemer	Projectmanager	Rapportage	Email, bellen	Wekelijks
Markt specialist	Projectmanager	Adviseren	Mondeling	Wekelijks/ dagelijks
Projectmanager	Stuurgroep	Informeren	Email, mondeling	Wekelijks
Projectmanager	Supervisor Fontys	Rapportage	Email, mondeling	Maandelijks
Projectmanager	Alle betrokkenen	informeren	Email, bellen	Bij onvoorziene omstandigheden

Bijlage B: Planning




2-4-2012
9-4-2012
16-4-2012
23-4-2012
30-4-2012
7-5-2012
14-5-2012
21-5-2012
28-5-2012
4-6-2012
11-6-2012

Bijlage C: Initiële Business Case

Inleiding

Dit document bevat de overwegingen om het huidige project te starten. Het vormt de rechtvaardiging van het project en zal om deze reden worden beoordeeld door de stuurgroep. De onderbouwing van het project zal regelmatig worden geëvalueerd op basis van deze Business Case. Het beschrijft de verwachte te investeren kosten ten opzichte van de verwachte voordelen en besparingen en de projectrisico's. De kosten/baten-afweging wordt onderbouwd middels een investeringsvoorstel.

Redenen

Capital Intervest onderscheidt zich van andere ontwikkelaars voor de beleggingsmarkten, door software op maat te maken en door over zeer veel koersdata te beschikken. Dit laatste wordt steeds lastiger, omdat er steeds meer bronnen komen die koersdata versturen. Die bronnen hanteren allemaal een eigen formaat om die koersdata te versturen. Hier wil Capital Intervest meer aandacht aan gaan besteden, zonder veel te complexe handelingen.

Voordelen

Capital Intervest zal na het uitvoeren van het onderzoek, een beter beeld krijgen over het koppelen van het .Net 4.0 Framework (waarmee ze werken) met oudere frameworks, in dit geval met ActiveX.
Met de uitkomsten van het onderzoek en de applicatie, zal de hoeveelheid koersdata die Capital Intervest te bieden heeft snel groeien. Hierdoor zullen klanten een betere binding hebben met het bedrijf. Verder zal het klanten aandeel van Capital Intervest ook groeien, omdat ze een unieke positie hebben ten opzichte van andere bedrijven in de beleggingsmarkt.

Tijd en Kosten

Het totale project heeft een doorlooptijd van 17 weken. Het aantal uren tijdens deze 17 weken zal in totaal ongeveer 600 uur zijn.
De financiering voor dit project zal relatief laag zijn. De kosten zijn in twee aspecten te onderscheiden:

- 1800 euro aan loon voor de projectmanager
- Tijd dat de andere werknemers kwijt zijn aan het begeleiden van de projectmanager

Risico's

Dit project brengt geen grote investering met zich mee en zal daarom financieel gezien weinig risico’s hebben.
Er is een kans dat de reputatie geschaad wordt, als het product in de praktijk onjuist werkt.

Investeringsvoorstel

De investering die Capital Intervest doet is tijd en geld. Tijd in de zin van dat er tijd over het project heen gaat voordat het af is, en tijd dat andere werknemers steken in het project.

Verder kost de afstudeerder geld, dat is dus de investering in geld. Dit ligt natuurlijk vele malen lager dan dat een werknemer het project zou moeten uitvoeren. Hierdoor liggen de investeringen dus laag.

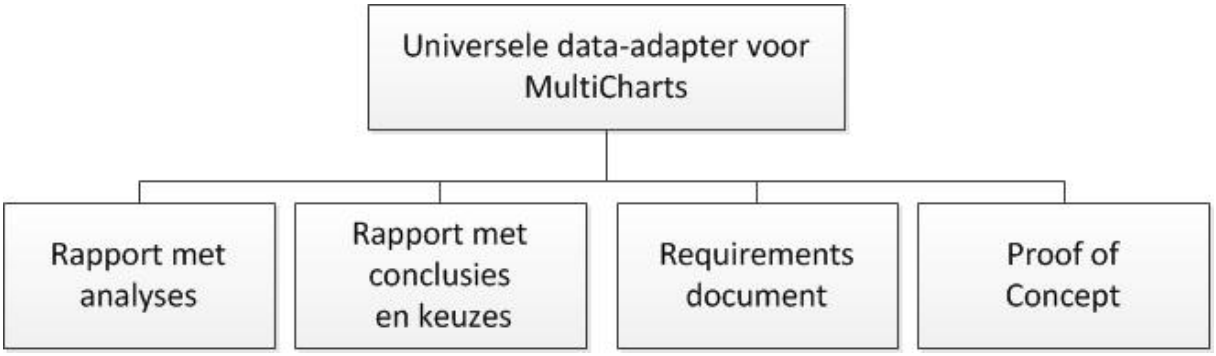
Verder zouden de resultaten in de toekomst geld kunnen opleveren waardoor de investeringen snel terug verdiend worden.

Bijlage D: Initieel Projectplan

Inleiding

In dit Initieel Projectplan staat vastgelegd in een paar korte stappen welke producten er worden opgeleverd. Dit wordt naast een beschrijving ook met een verloop duidelijk gemaakt. Het diagram dient u dan ook van links naar rechts te lezen.

Producten



Rapport met analyses

Het rapport met analyses is het eerste document wat er wordt opgeleverd. In dit document staat:

- de roadmap van Capital Intervest
- API van MultiCharts
- Informatie ActiveX koppeling met .Net

Rapport met conclusies en keuzes

Het rapport met conclusies en keuzes bevat de overwegingen en de daaruit volgende keuzes die gemaakt zijn, over hoe het Proof of Concept moet worden gemaakt. Dit rapport wordt met behulp van het rapport met analyses gemaakt.

Requirements document

In dit document komen de eisen van de opdracht gever terug. Verder worden de conclusies en keuzes uitgewerkt en wordt er een ontwerp gemaakt voor het Proof of Concept.

Proof of Concept

Het Proof of Concept is de realisatie van het Requirements document. Het Proof of Concept zal moeten bewijzen dat het rapport met conclusies en keuzes gerechtvaardigd wordt en zal zich uiten in een applicatie.

Universele data-adapter voor MultiCharts

Dit is de verzameling van de rapporten, het Requirements document, de applicatie en de documentatie.

Bijlage II

Roadmap CI

Roadmap CI

Waar wil CI naar toe met de universele data-adapter

Beschrijving over waar Capital Intervest naar toe wil met het afstudeerproject. Wat zijn de mogelijkheden, knelpunten en wat kan er verbeterd worden?

28-2-2012

Inhoudsopgave

1. Inleiding

Aan de hand van een interview met de bedrijfsbegeleider is het duidelijk geworden wie Capital Intervest is en waar ze voor staan. Er is een concreet beeld gevormd waar Capital Intervest met dit project naar toe wil. Er wordt ook met de bedrijfsbegeleider teruggekoppeld voor een definitief project.

Capital Intervest wil graag een unieke positie hebben in de beleggingsmarkt en daarvoor moet er goed op de markt ingespeeld worden. Er lopen al diverse projecten bij Capital Intervest om innovatief en uniek te blijven.

Voor het afstudeerproject dat ik ga uitvoeren, zijn er nog een aantal zaken die nog worden uitgevoerd. Er wordt duidelijk in kaart gebracht wat het probleem is, hoe het opgelost gaat worden en hoe Capital Intervest daar naar toe wilt werken. Verder wordt de huidige situatie van Capital Intervest belicht en waar het in de toekomst wil gaan staan. Tenslotte wordt nog kort belicht waar de oplossing uit bestaat.

De koersen op de beurzen zijn constant in beweging. Kost een aandeel van bedrijf X €5,-, kan dat zelfde aandeel enkele seconden later alweer €6,90 kosten. Het is dus van belang dat er snel gehandeld wordt. Dit geld niet alleen voor aandelen, maar ook op alle financiële producten die te verhandelen zijn.

Deze snelheden kunnen alleen bereikt worden door beleggers die op professioneel niveau handelen. En zelfs zij kunnen te laat reageren op de koerswisselingen. Laat staan hoe erg de particuliere beleggers achter de feiten aanlopen.

Een oplossing hiervoor, die al veel gebruikt wordt, is het automatiseren van orders plaatsen. Dit biedt aan zowel de professionele, als de particuliere belegger, de mogelijkheid om op top snelheden orders te plaatsen.

Echter blijft er nog een probleem over, namelijk de verschillende typen koers-data. Deze ontstaan doordat er steeds meer bronnen komen die koers-data leveren in hun eigen protocol. Niet alle software pakketten kunnen om gaan met al die verschillende type koersdata.

Hiervoor komt er een data-adapter, die alle verschillende koers-data omzet naar één formaat, hiermee is het mogelijk te gebruiken bij meerdere software pakketten.

2. Huidige situatie

2.1. Visie en missie

"Capital Intervest is een onafhankelijke aanbieder van markt leidende, innovatieve producten en diensten voor de wereldwijde beleggersmarkt. Een ongekend uitgebreide range van financiële markten en producten is beschikbaar voor iedere serieuze belegger via 'top of the bill' handelsplatformen of op maat gemaakte applicaties en handelsomgevingen speciaal ontwikkeld voor zowel de particuliere als professionele belegger."

Bron: http://www.capitalintervest.nl/particulier/over-capital-intervest

Capital Intervest onderscheidt zich dus door software op maat te maken voor de beleggers. Hierbij willen ze graag ook koppelingen maken met software pakketten van derden.

2.2. Stand van zaken

Capital Intervest beschikt nu over een softwarepakket genaamd TradeRouter. Deze applicatie is in staat om handelsadviezen op te pikken en daarop orders te plaatsen en door te sturen naar softwarepakketten die orders kunnen uitvoeren. Er ontbreekt echter een koppeling tussen TradeRouter en MultiCharts.

MultiCharts is een applicatie die handelsadviezen genereert aan de hand van analyses uit koers verlopen. Men kan met deze handelsadviezen weer een keuze maken om te gaan beleggen op de beurs. Deze handelsadviezen kunnen ook als handelssignalen worden gebruikt. TradeRouter maakt gebruik van handelssignalen om een order uit te voeren.

Er zijn echter meerdere software pakketten zoals TradeRouter, die ook wel brokerapplicaties worden genoemd. Deze leveren de koersdata allemaal in een eigen formaat, wat het weer lastig maakt om te gebruiken in MultiCharts.

2.3. Vooruitzicht

Wanneer de tussenlaag tot stand is gekomen, is het mogelijk om diezelfde tussen laag voor meerdere software pakketten te gaan gebruiken. Tevens kan deze gebruikt gaan worden in producten die in de toekomst worden ontwikkeld door Capital Intervest. Er kunnen dus projecten ontstaan door de resultaten van dit project.

Het kan zijn dat Capital Intervest met een nieuwe handelsapplicatie komt. Het zou dan ideaal zijn om die dan ook gelijk aan MultiCharts te koppelen.

Een ander idee is om een dataleverancier te koppelen met TradeRouter. Die koersdata kan dan via de TradeRouter door gestuurd kunnen worden naar MultiCharts.

Veel kansen en mogelijkheden dus voor Capital Intervest.

3. Doel

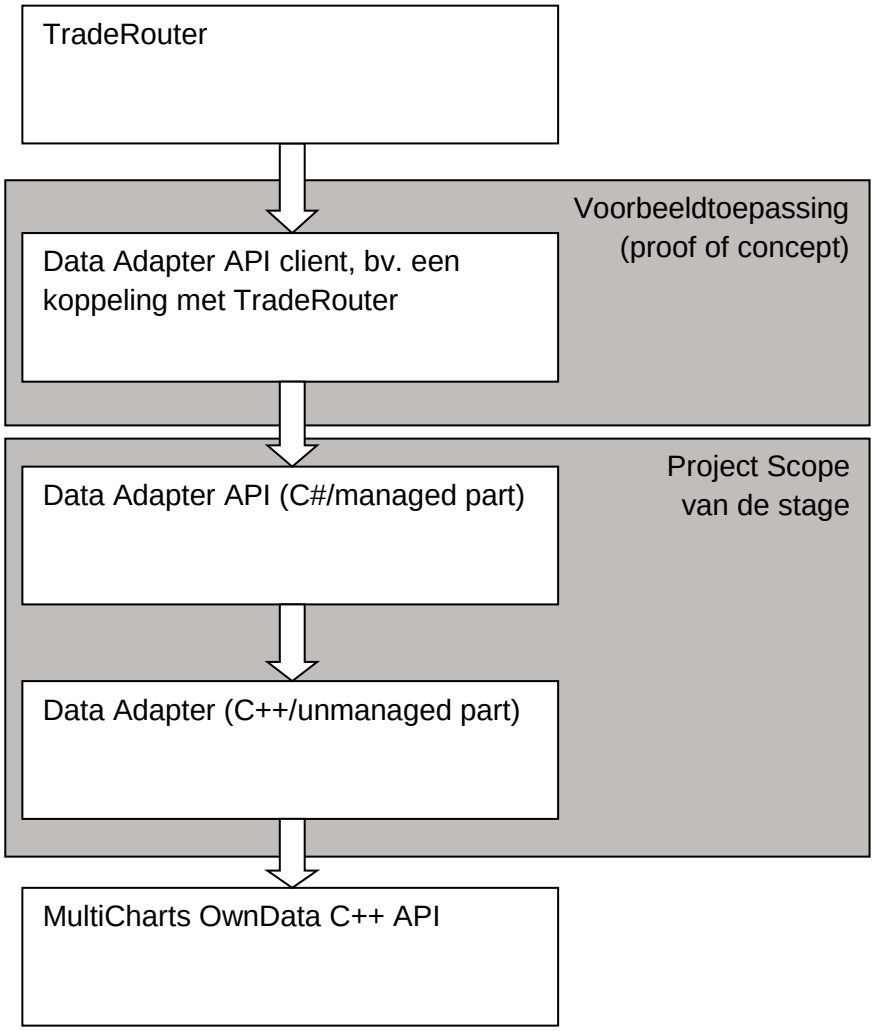
3.1. Waarom?

MultiCharts is een product van een derde partij. Vandaar dat er nog geen koppeling is tussen MultiCharts en TradeRouter. TradeRouter is het product van Capital Intervest en maakt gebruik van handelsadviezen die via signalen binnen komen. MultiCharts kan deze signalen generen en is dus een uitbreiding voor TradeRouter, waarmee de klanten dus meer kunnen gaan handelen.

3.2. Product

Waar Capital Intervest naar toe wil, is om aan hun klanten MultiCharts aan te bieden en gebruikers die MultiCharts al gebruiken, de mogelijkheid geven om via TradeRouter geautomatiseerd te handelen.

Niet alle brokers hebben een koppeling met MultiCharts. Met de koppeling tussen TradeRouter en MultiCharts worden dit er meer. Hiermee geeft Capital Intervest nog meer mogelijkheden tot handelen. Het biedt de gebruikers van MultiCharts meer koersdata. Hieronder volgt een simpel voorbeeld van de uitgangssituatie.



TradeRouter is in dit geval in één van de brokers. Het onderste beige stuk is het universele stukje software. Het Proof of Concept is dan de koppeling tussen de broker en de Data Adapter, die weer gekoppeld is met de API van MultiCharts. De volgende punten zullen hiermee verwezenlijkt worden:

- Meer koersdata voor MultiCharts
- MultiCharts is te gebruiken voor TradeRouter
- TradeRouter ontvangt meer signalen
- Makkelijker koppelingen maken tussen MultiCharts en andere brokers

Het is dus de bedoeling dat de koppeling tussen de broker en de Data Adapter eenvoudig is te realiseren. Dit zou gedemonstreerd kunnen worden door een andere programmeur een koppeling te laten maken met een andere broker.

3.3. Onderzoek

Waar het vooral om draait in dit project, is het onderzoek. Het onderzoek ontstaat uit een onderzoeksvraag. Deze onderzoeksvraag luid: "Is het haalbaar is om een efficiënte, betrouwbare en universele data-adapter te bouwen om data aan te leveren aan MultiCharts?".

Om de onderzoeksvraag goed te kunnen beantwoorden is het van belang dat er een onderzoek wordt gedaan naar de mogelijkheden. De gevonden mogelijkheden en de afweging van deze mogelijkheden worden in het document Unmanaged naar Managed gepresenteerd.

In het PID zijn er nog een aantal deelvragen te vinden. Deze zullen beantwoord worden of in het Unmanaged naar Managed document of in de scriptie.

De softwareproducten zullen uiteindelijk de antwoorden van het onderzoek bevestigen. Dit wordt ook wel genoemd als 'Proof of Concept'.

3.4. Tijdsbestek

Capital Intervest wil graag op 22-06-2012 de Data Adapter API af hebben. Dit wordt gedemonstreerd door een koppeling te maken tussen de Data Adapter API en TradeRouter.

Tussen door worden er nog een aantal tussenproducten gemaakt. Hierbij kunt u denken aan documenten, applicaties en presentaties.

Eerst zal er documentatie gemaakt worden. Hier onder valt het volgende:

- PID
- Roadmap CI
- Onderzoek Unmanaged naar Managed
- Requirements document

Dit allemaal moet binnen 8 weken vanaf de start van het afstuderen af zijn. Dit komt neer op vrijdag 13 april 2012. Er wordt gewerkt in de volgorde van het lijstje.

Nadat de documentatie af is, kan er worden gewerkt aan de software. Hiervoor worden de volgende tussenproducten gemaakt:

- MultiCharts API Managed gedeelte
- Data ontvangen van TradeRouter
- Managed API koppelen met TradeRouter

Zoals eerder vermeld moet de Data Adapter API af zijn op 22-06-2012. De volgorde van het lijstje hierboven is ook de volgorde van de stukken software. Voor deze tussenproducten is er dus 11 weken uitgetrokken.

Bijlage III

Unmanaged naar Managed

Unmanaged naar Managed

Hoe managed C# aan unmanaged C++ wordt gekoppeld

Er zijn veel redenen voor het koppelen van C# met C++. De voornaamste reden is dat C# een moderne taal is met steeds meer mogelijkheden. Er zijn meerdere manieren van het koppelen van C# met C++. Ze worden allemaal behandeld en één daarvan wordt er gekozen.

28-2-2012

Inhoudsopgave

1. Inleiding

Om verder te borduren op het tienstappenplan, is het nu tijd voor het diepteonderzoek. Met de resultaten van het diepteonderzoek kan er gewerkt worden aan het oplossingsplan. Er dient echter wel een goed onderzoek worden uitgevoerd, zodat het oplossingsplan ook daadwerkelijk een oplossing vormt.

In de IT wereld worden oudere programmeertalen en technieken niet zomaar opgegeven. Zeker niet in het geval van C++. C++ is platformonafhankelijk en qua performance één van de snelste programmeertalen. Dit zijn de redenen waarom er nog veel met C++ gewerkt wordt, ondanks dat het lastiger is om mee te werken dan modernere programmeertalen, zoals C#.

Vooral performance is van belang voor Capital Intervest, omdat de koersdata zo snel mogelijk op het scherm van de gebruiker moet worden weergeven. Er wordt dan ook veel aandacht besteed aan performancebewust programmeren.

Voor de Data Adapter moet er eerst wat onderzoek gedaan worden tussen de verschillende methodes om C# te kunnen koppelen met C++, om zo probleemloos de API te kunnen gebruiken in de .Net omgeving.

De API van MultiCharts werkt met ActiveX elementen, geschreven in C++. Dit is een wat verouderde techniek, maar wordt tot de dag van vandaag nog gebruikt door veel systemen. De API van MultiCharts is daar één van.

Omdat ActiveX in C++ ontwikkeld is, bevindt het zich in een unmanaged omgeving. Dit houdt in dat bij het programmeren, de programmeur rekening moet houden met het geheugenbeheer. Dit vraagt voor meer zorg en oplettendheid van de programmeur. Er is dus meer kans op fouten. Veel fouten zijn niet meteen te zien, maar komen pas aan het licht tijdens de uitvoering van de applicatie. Als dit het geval is, dan worden de fouten hersteld. Vervolgens moet er weer getest worden. Dir zorgt ervoor dat het proces vertraging op loopt.

Ook zijn er minder libraries beschikbaar voor C++, en moet de programmeur dus meer zelf uitprogrammeren. Hierdoor is de programmeur meer tijd kwijt aan het programmeren van simpele code, die waarschijnlijk al een keer eerder is geprogrammeerd.

Dit allemaal zijn redenen om een onderzoek te doen om een laag te maken in C#, die de API aanspreekt van MultiCharts in C++. In de volgende hoofdstukken worden de verschillende methoden besproken en uiteindelijk een keuze gemaakt.

2. Unmanaged C++ aanroepen vanuit C#

2.1 Uitleg

Het idee van deze methode is om een .dll bestand te maken van een C++ project. Hierbij worden er geen classes gebruikt, maar alleen functies. Deze functies zijn niet gekoppeld aan een class en zullen dus alleen data kunnen ontvangen en doorsturen (eventueel ook weg schrijven). In het geval van een laag tussen de API van MultiCharts en een broker, is dit voldoende.

Extra uitleg: http://snippets.dzone.com/posts/show/7375

2.2 Voordelen

De .dll zou uiteindelijk alleen uit een header en .cpp bestaan. Hiermee is alles terug te vinden in twee bestanden.

Cpp.h

```
extern "C" __declspec(dllexport)
                          int __stdcall add(int a, int b);
```

Cpp.cpp

```
#include "cppTestDLL.h"

int __stdcall add(int a, int b)
{
        return a + b;
}
```

In C# is het mogelijk om in één class of meerdere, zelf te bepalen welke methoden uit de .dll gebruikt kunnen worden.

2.3 Nadelen

Dat alles alleen in een header bestand en .cpp bestand staat, kan juist ook een nadeel zijn. Als het project groot is, kan het onoverzichtelijk worden om code terug te vinden. Tevens is dit niet object-georiënteerd programmeren.

Een ander nadeel is dat er in het C# project voor iedere methode in het .dll bestand, een stukje code moeten worden geschreven over hoe de methode eruit ziet in de .dll. Dit vereist dat de gebruiker in principe iedere methode in C++ moet uitschrijven en in C# de signatuur moet worden geschreven.

```
using System.Runtime.InteropServices;

namespace ManagedMain
{
        public class Cpp
        {
                [DllImport(
                        "Cpp.dll",
                        EntryPoint = "add",
                        ExactSpelling = false
                        )] // Imports the CPP DLL
                public static extern Int32 add(Int32 a, Int32 b);
        }
}
```

De bovenstaande stukken code zijn te vinden op: http://snippets.dzone.com/posts/show/7375

3. ActiveX objecten maken met C#

3.1 Uitleg

C# behoord tot het .Net Framework en kan overweg met COM technologie, ActiveX onderdeel van is. Dit houdt in dat er met C# een ActiveX component kan worden gemaakt. De API van MultiCharts werkt ook met ActiveX componenten.

C# code wordt omgezet in een ActiveX component, zodat er gemakkelijk gecommuniceerd kan worden met de API van MultiCharts.

Extra uitleg: http://www.dreamincode.net/forums/topic/38890-activex-with-c%23/

3.2 Voordelen

Er kan dus in C# gecodeerd worden zonder ervoor iets te hoeven in te leveren.

Gemakkelijke integratie met de MultiCharts API. Die is namelijk in C++ met ActiveX geschreven, waardoor er geen tussenlagen gemaakt hoeven te worden.

Omdat beide delen met ActiveX componenten werken, kan er gemakkelijk gecommuniceerd worden.

3.3 Nadelen

Er moet rekening worden gehouden met ActiveX stijlen. Dit houdt in dat er code moet worden geschreven die vereist is voor ActiveX. Dit zorgt voor extra werk. Een voorbeeld hiervan is te zien in het stukje code. De code boven de class is specifiek voor ActiveX.

Tenslotte gebruik je dan niet de API in C#, maar ga je op deze manier de C# code in de ActiveX C++ omgeving gebruiken. Hierdoor werk je uiteindelijk ook weer in C++.

```
class Hello
   {
       // A very simple interface to test ActiveX with.
       [
           Guid("E86A9038-368D-4e8f-B389-FDEF38935B2F"),
           InterfaceType(ComInterfaceType.InterfaceIsDual),
           ComVisible(true)
       ]
       public interface IHello
       {
           [DispId(1)]
           string Hello();

           [DispId(2)]
           int ShowDialog(string msg);
       };
       [
          Guid("873355E1-2D0D-476f-9BEF-C7E645024C32"),
           // This is basically the programmer friendly name
           // for the guid above. We define this because it will
           // be used to instantiate this class. I think this can be
           // whatever you want. Generally it is
           // [assemblyname].[classname]
          ProgId("csharpAx.CHello"),
           // No class interface is generated for this class and
           // no interface is marked as the default.
           // Users are expected to expose functionality through
           // interfaces that will be explicitly exposed by the object
           // This means the object can only expose interfaces we define
          ClassInterface(ClassInterfaceType.None),
           // Set the default COM interface that will be used for
           // Automation. Languages like: C#, C++ and VB
           // allow to query for interface's we're interested in
           // but Automation only aware languages like javascript do
           // not allow to query interface(s) and create only the
           // default one
          ComDefaultInterface(typeof(IHello)),
          ComVisible(true)
      ]
       public class CHello : IHello
       {
           #region [IHello implementation]
           public string Hello()
           {
               return "Hello from CHello object";
           }
           public int ShowDialog(string msg)
           {
               System.Windows.Forms.MessageBox.Show(msg, "");
               return 0;
           }
           #endregion
       };
   }
```

Deze code is te vinden op: http://www.dreamincode.net/forums/topic/38890-activex-with-c%23/

4. .dll bestand vanuit C++in C#

4.1 Uitleg

In C# kun je .dll bestanden importeren en gebruiken. Het idee is dat je in C++ managed code gaat schrijven. Dit gebeurt door middel van een aantal lagen waarvan één van de lagen beschikbaar wordt in C#. Uiteindelijk zal de laag die in C# beschikbaar is een of meerdere classes bevatten die te gebruiken zijn in C#, alsof ze in C# geschreven zijn. Dit gebeurd door het C++ bestand te compileren naar een .dll bestand en die in C# te importeren.

Extra info: http://blogs.msdn.com/b/borisj/archive/2006/09/28/769708.aspx

4.2 Voordelen

De C++ managed class kan in C# gebruikt worden alsof het een C# class is. Dit maakt het gemakkelijker om het in C# te gebruiken.

C++ code

```
//cppcliwrapper.cpp
#include "cppcliwrapper.h"
#include "marshal.h"

using namespace cppcliwrapper;

ManagedHelloWorld::ManagedHelloWorld() : hw(new HelloWorld())
{
}

ManagedHelloWorld::~ManagedHelloWorld()
{
      delete hw;
}

void ManagedHelloWorld::SayThis(System::String^ phrase)
{
      hw->SayThis(marshal::to<wchar_t*>(phrase));
}
```

C# code

```
using System;
using System.Text;
using cppcliwrapper;

namespace CSharpDirectCaller
{
    class Program
    {
        static void Main(string[] args)
        {
            ManagedHelloWorld mhw = new ManagedHelloWorld();
            mhw.SayThis("I'm a C# application calling native C++ code!");
        }
    }
}
```

Er hoeft alleen maar een class geschreven te worden in C++ die managed is, die dan gebruikt wordt in C#. Het natuurlijk ook mogelijk om meerdere classes hiervoor te gebruiken.

4.3 Nadelen

Je moet de data die overgestuurd wordt omzetten naar bruikbare data voor zowel C++ als voor C#. Dit heet marshalling. Managed data wordt dan omgezet naar een pointer en vice versa.

```
   namespace marshal {
      template <typename T>
      static T to(System::String^ str)
      {
      }

      template<>
      static wchar_t* to(System::String^ str)
      {
            pin_ptr<const wchar_t> cpwc = PtrToStringChars(str);
            int len = str->Length + 1;
            wchar_t* pwc = new wchar_t[len];
            wcscpy_s(pwc, len, cpwc);
            return pwc;
      }
    }
```

De bovenstaande stukken code zijn te vinden op:
http://blogs.msdn.com/b/borisj/archive/2006/09/28/769708.aspx

5. Interfaces aanbieden aan C++ en C#

5.1 Uitleg

In C# wordt er een interface gedefinieerd en een class gemaakt die de interface implementeert. In C++ wordt er een zogenaamde Type Library (.tlb) bestand geïmporteerd. Verder wordt er in C++ er een class gemaakt die een instantie maakt van de C# interface en worden de methodes ingevuld. In C++ worden de API classes geëxporteerd in een .dll bestand. Zo zijn de API classes in C# te gebruiken.

5.2 Voordelen

Op deze manier hoef je geen datatypen om te zetten van C++ naar C# types en vice versa. Dit scheelt conversies.

Omdat je ook de interfaces en classes in beide omgevingen kunt aanroepen, is het gemakkelijker om de methodes in beide omgevingen te implementeren.

C++ code

```
#import "..\CsharpPart\bin\Release\CsharpPart.tlb" raw_interfaces_only
using namespace CsharpPart;

class ManagedDLLInterface
{
protected:
        IManagedMultiChartsAPIPtr pIMC;
        void InitInterface()
        {
                if (pIMC == NULL)
                {

        pIMC = new IManagedMultiChartsAPIPtr( __uuidof(ManagedMultiChartsAPI));
                }
        }

public:
        BSTR get_Name()
        {
                BSTR strResult;
                InitInterface();

                pIMC->getName(&strResult);
                return strResult;
        }
…
```

C# code

```
 public interface IManagedMultiChartsAPI
    {

        string getName();

        string getShortName();
…
```

In C# maak je een interface die de zelfde methodes heeft als de class in C++. Vervolgens maak je een class in C# die de methodes implementeert. In C++ importeer je een TLB bestand van het C# gedeelte. Zo kun je een object maken van de interface en gebruiken in C++.

5.3 Nadelen

Je bent verplicht om in beide omgevingen de methode te definiëren en te koppelen. Dit betekent niet dat de methode twee uitgeschreven hoeft te worden.

C++ code

```
// Import the type library.
#import "..\CsharpPart\bin\Release\CsharpPart.tlb" raw_interfaces_only
using namespace CsharpPart;
class ManagedDLLInterface
{
protected:
        IManagedMultiChartsAPI pIMC;
        void InitInterface()
        {
                if (pIMC == NULL)
                {
        pIMC = new IManagedMultiChartsAPI( __uuidof(ManagedMultiChartsAPI));
                }
        }
public:
        ManagedDLLInterface()
        {
                // Initialize COM.
                HRESULT hr = CoInitialize(NULL);
                pIMC = NULL;
        }
        ~ManagedDLLInterface()
        {
                pIMC = NULL;
                // Uninitialize COM.
                CoUninitialize();
        }
        BSTR get_Name()
        {
                BSTR strResult;
                InitInterface();
                pIMC->get_Name(&strResult);
                return strResult;
        }
```

De bovenstaande stukken code komen uit een project die door Van Der Moolen is geschreven.

6. Conclusie

Om een keuze te maken uit de drie methodes, moeten ze vergeleken worden en voor- en nadelen afwegen.

	Hoeveelheid program-meer werk in C#	Hoeveelheid program-meer werk in C++	'Dubbele' code	Tussen lagen	Managed code	Object georiën-teerd program-meren
Unmanged C++ aanroepen vanuit C#	+	+/-	--	++	+	--
ActiveX object maken met C#	+	--	+	+	-	+/-
.dll bestand vanuit C++ in C#	++	+/-	++	+/-	++	++
Interfaces aanbieden aan C++ en C#	+	+	+	++	++	++

- -- = zeer slecht
- - = slecht
- +/- = noch slecht, noch goed
- + = goed
- ++ = zeer goed

Hoeveelheid programmeer werk in C#:
Als de hoeveelheid programmeer werk meer is in C#, dan valt de score hoger uit.

Hoeveelheid programmeer werk in C++:
Als de hoeveelheid programmeer werk meer is in C++, dan valt de score lager uit.

'Dubbele' code
Met 'Dubbele' code wordt bedoeld of de methodes in zowel C++ als in C# gedefinieerd moeten worden.

Tussen lagen
Hoe meer tussen lagen moeten worden toegevoegd, hoe lager de score.

Managed code
Als er meer aandacht besteedt moet worden aan geheugen beheer, dan is de code minder managed, dus valt de score lager uit.

Object georiënteerd programmeren
Als de code zich in meerdere bestanden bevind en er met classes wordt gewerkt, dan kunnen we van object georiënteerd programmeren spreken.

Na deze onderdelen te hebben vergeleken tussen de vier methodes, is de 'Interfaces aanbieden aan C++ en C#' als beste keus naar voren gekomen. Het grote voordeel ten opzichte van 'dll bestand vanuit C++ in C#', is dat marshalling van de datatypen is niet nodig is. Dit scheelt wat programmeerwerk en vermindert de kans op fouten met conversies van datatypen. Een ander voordeel is dat classes en interfaces van C# te gebruiken zijn in C++ en vice versa. Dit maakt het koppelen gemakkelijk.

Bijlage IV

Requirements data adapter

Data-adapter Requirements

Eisen voor de software

Hoe de verschillende onderdelen van het project er komen uit te zien. Ze moeten aan bepaalde eisen voldoen en volgens een bepaalde architectuur gebouwd worden.

13-3-2012

Inhoudsopgave

1. Inleiding

Na het diepteonderzoek te hebben doorlopen, is het nu tijd voor het oplossingsplan. Met de resultaten van het diepteonderzoek wordt het oplossingsplan gemaakt. Nadat het oplossingsplan is gevormd, wordt het met de bedrijfsbegeleider besproken.

Het koppelen van TradeRouter met MultiCharts gebeurd door middel van het maken van lagen in de software en die met elkaar te verbinden. Via deze lagen kan TradeRouter koersdata leveren aan MultiCharts. MultiCharts analyseert deze aangeleverde koersdata en genereert handelsadviezen. Deze handelsadviezen worden weer gestuurd naar TradeRouter die er dan weer mee automatisch gaat handelen.

Dit is in grote lijnen de opzet van de software. Met deze opzet wordt er rekening gehouden met:

- Wijzigingen
- Onderhoud
- Vernieuwingen

Er zijn drie lagen aanwezig in het project:

- MultiCharts OwnData API C++
- MultiCharts API C#
- Koppeling tussen TradeRouter en MultiCharts API in C#

Per laag wordt er uitgelegd wat zijn rol is in het geheel, wat de kenmerken zijn en hoe het gebruikt wordt.

Zijn ook nog bepaalde eisen waaraan moeten worden voldaan vanuit de opdrachtgever. Deze eisen moeten worden verwerkt in het project om het slagen te verzekeren.

Om het implementatiegedeelte van het project zo soepel mogelijk te laten verlopen, moet er eerst goed voorbereid worden. Het helpt om de lagen nader te bestuderen en goed uit te schrijven, zodat het duidelijk wordt hoe deze moeten worden geïmplementeerd.

2. Lagen structuur

2.1 Voordelen

Tegenwoordig wordt software vaak in lagen geschreven. Deze vorm van software architectuur wordt ook wel Multilagenarchitectuur genoemd. Een veel voorkomende vorm van deze architectuur is een 3-lagenachitectuur. Hierin is er een afzonderlijke laag voor de presentatie, business logica en datatoegang/dataopslag. Het is mogelijk om nog meer lagen in de architectuur te werken, om zo iedere laag een specifieke rol te geven. Deze architectuur heeft vele voordelen te bieden aan programmeurs.

Doordat de software is opgedeeld in lagen, is het ook weer mogelijk om functionaliteit te ontkoppelen en te vervangen door een andere laag in te zetten. De lagen worden aan elkaar gekoppeld en werken zo als een geheel.

Wat er ook nog bij komt kijken is dat wanneer er een wijziging moet plaats vinden in een bepaalde laag, dat de andere lagen hier niets van merken. Zo is het gemakkelijk om per laag te werken, zonder de andere lagen te beïnvloeden. Dit maakt het opsporen van fouten ook eenvoudiger, omdat de programmeur weet in welke laag hij als laatste aanpassingen heeft verricht.

Tevens kan een laag worden hergebruikt in een ander systeem, waardoor er niets hoeft worden te herschreven. Een laag heeft dus zijn eigen verantwoordelijkheden en kan onafhankelijk ingezet worden in meerdere systemen.

Als laatste, maar niet het minste, is een zeer groot voordeel dat er beter parallel gewerkt kan worden. Functionaliteit kan immers opgedeeld worden in de verschillende lagen. Omdat iedere laag afzonderlijk kan werken, hoeven de programmeurs ook niet op elkaar te wachten om te testen. Dit verhoogt de productiviteit en ook kwaliteit van het werk, omdat een programmeur goed bekend is met de laag waarin hij heeft gewerkt.

Dit dient niet verward te worden met een component based architectuur. Een belangrijk verschil is dat een laag een stuk software is dat onafhankelijk kan draaien. Een component kan niet alleen draaien, maar moet echt ingezet worden in een groter geheel. Er is nog een verschil tussen lagen architectuur en component based architectuur, die vooral duidelijk is tijdens het programmeren. Bij een lagen architectuur hoeft de programmeur niets te weten van de andere lagen, alleen de interfaces implementeren. Bij een component architectuur, dient de programmeur wat meer te weten dan alleen de interface van de andere component. Bijv. ook het formaat waarin de data wordt verwerkt.

2.2 In het project

In het project wordt er onderscheid gemaakt in drie lagen:

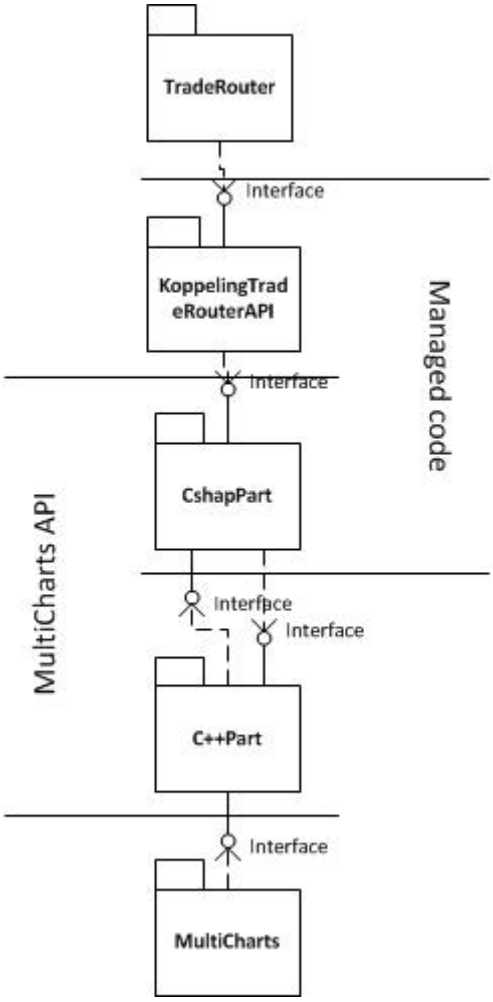
- MultiCharts OwnData API C++
- MultiCharts API C#
- Koppeling tussen TradeRouter en MultiCharts API C#

De eerste twee lagen zullen nauw met elkaar samen werken en vormen samen de API van MultiCharts.

De MultiCharts OwnData API in C++ wordt aangeleverd door MultiCharts Inc. De API wordt in source code aangeleverd. De code gebruik je dan in je eigen project en hoeft verder niet aangepast te worden.

De MultiCharts API in C# wordt boven de laag in C++ gebouwd, zodat er van buiten af via C# benaderd kan worden. Deze laag zorgt er dus voor de dat de unmanaged data vanuit de C++ laag omgezet word naar managed data. Vanuit het C# gedeelte is het gemakkelijk om verder met de data te werken omdat het managed is.

De koppeling tussen TradeRouter en de MultiCharts API in C#, maakt in principe gebruik van de API. Voor deze laag maakt het niet uit of het in C++ of C# geschreven is, als het maar de API benadert en de data heen en terug kan sturen.



Deze koppeling zal dus alleen tussen TradeRouter en de MultiCharts API werken. Om de MultiCharts API met een andere applicatie te laten werken, moet er een andere koppeling ontwikkeld worden. De twee andere lagen kunnen worden hergebruikt. Dit is dus de kracht van de lagenarchitectuur.

3. Requirements

De opdrachtgever heeft bepaalde eisen opgesteld. Om het project goed te laten beoordelen, moet er worden voldaan aan de deze eisen. Als deze er niet in verwerkt zitten, dan zal de opdracht gever niet tevreden zijn met het product en het project met een onvoldoende te beoordelen.

3.1 Niet-functionele eisen

Betrouwbaarheid. Hiermee wilt de opdrachtgever zeggen dat de lagen altijd verbindingen met elkaar moeten hebben en met de applicatie waarmee ze werken. Deze eis komt voort uit dat koersdata binnen een seconde alweer verouderd is. Mocht er enkele seconden geen verbinding zijn, dan mist de gebruiker al waardevolle informatie. Hierdoor kan de gebruiker zonder het te weten handelen met verouderde koersdata en is dat nadelig voor de gebruiker. Hierdoor zal de gebruiker de applicatie als onbetrouwbaar beschouwen.

Efficiënt. De applicatie moet efficiënt om kunnen gaan met de koersdata die binnenkomt. De koersdata kan mogelijk meerdere malen per seconde binnenkomen. De gebruiker zal waarschijnlijk meerdere koersen volgen. Dit houdt in dat er vele koersdata tegelijkertijd binnenkomt en dat deze allemaal moet worden doorgestuurd. Het is dus van belang dat al deze data snel verwerkt wordt, zonder dat er data kwijt geraakt wordt. Anders komt de betrouwbaarheid in gevaar.

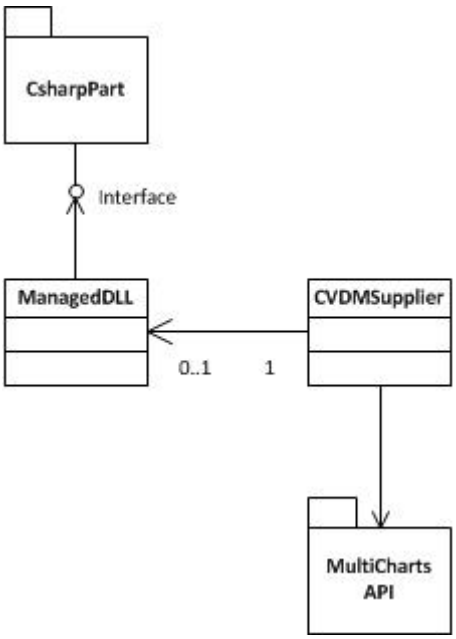
Flexibel inzetbaar. Dit wordt door middel van de multilagenarchitectuur verwezenlijkt. De verschillende lagen maakt het mogelijk, door een andere bron van koersinformatie gemakkelijk te kunnen koppelen. De data adapter is een interface voor MultiCharts. De data adapter is universeel voor de brokerapplicaties. Het is zodat de brokerapplicaties gemakkelijk gekoppeld kunnen worden aan MultiCharts. Het zal uiteindelijk gekoppeld worden aan de bovenste laag.

3.2 Functionele eisen

Een managed interface. De interface die vanuit de C# MultiCharts API wordt aangeboden moet dus managed zijn. Dit wordt al afgedekt, doordat de C# laag van de API een interface aanbiedt en C# een managed programmeertaal is.

4. MultiCharts OwnData API C++

4.1 Klassendiagram



Zoals te zien is, is het een vrij simpel klassendiagram. Dit komt omdat de kern van de MultiCharts API aangeleverd komt vanuit MultiCharts Inc. Dit pakket wordt verder beschreven in het OwnData API document dat als bijlage is toegevoegd.

Deze laag is al eerder gemaakt voor Van Der Moolen BV. Deze broker had deze laag oorspronkelijk laten bouwen om met een Visual Basic laag te koppelen. Van Der Moolen BV is inmiddels failliet, dus kan er niet verder om informatie worden gevraagd.

Pragmatix, het bedrijf achter de technologie van Van der Moolen BV. heeft de code geschreven voor deze laag. Ze zijn ook in bezit van de licenties. Pragmatix werkt nauw samen met Capital Intervest, waardoor deze software vrij gebruikt mag worden met toestemming van Pragmatix.

CVDMSupplier heeft alle methodes die gebruikt worden in de MultiCharts API. In de methodes wordt de data die vanuit de MangedDLL komt doorgegeven aan de MultiCharts API en vice versa. CVDMSupplier verbindt als het ware de MultiCharts API met het Managed gedeelte.

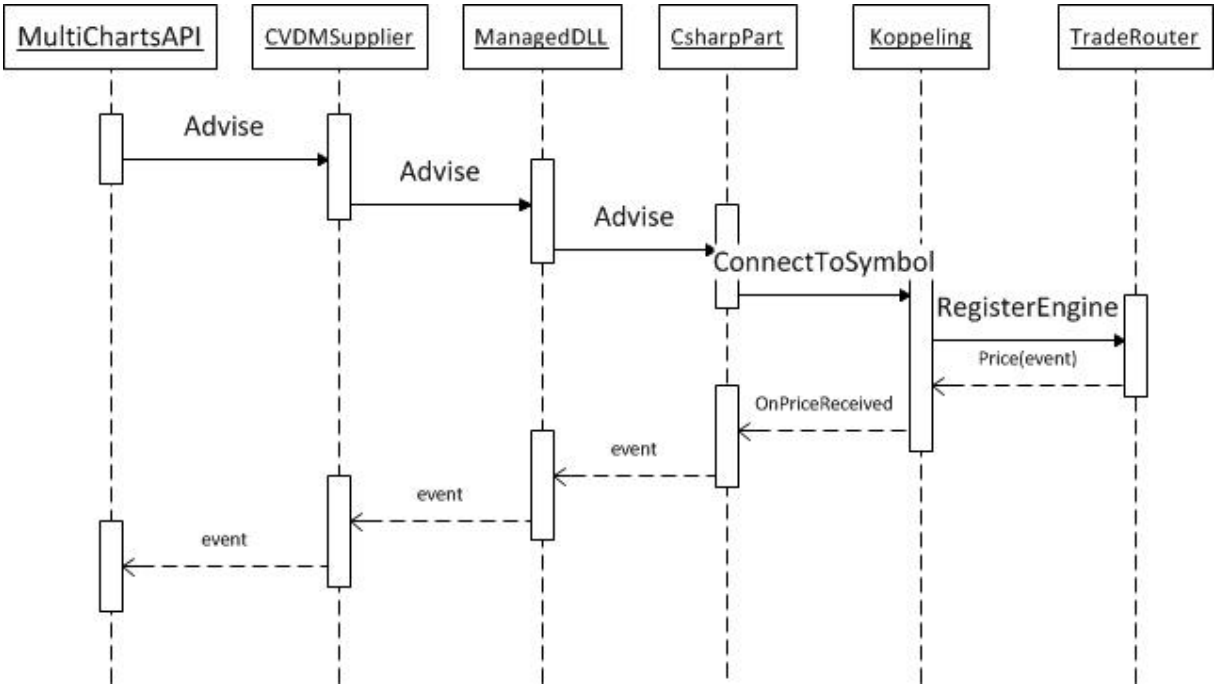
ManagedDLL zorgt ervoor dat de CsharpPart verzoeken binnen kunnen komen. Tevens maakt het de managed data unmanaged, zodat de MultiCharts API ermee overweg kan. Tenslotte zorgt de ManagedDLL dat de unmanaged data die vanuit de MultiCharts API komt, om wordt gezet naar managed data die weer door kan worden gegeven aan de CsharpPart.

4.2 Voorbeeld

4.2.1 Use Case

Naam	Voeg symbool stream toe
Primair actor	Gebruiker
Pre-Condities	TradeRouter draaiende hebben MultiCharts gekoppeld hebben met TradeRouter
Hoofd Succes Scenario	1. Gebruiker opent een nieuwe chart window in MultiCharts 2. Gebruiker voegt een instrument aan de chart window 3. Data komt nu binnen
Post-Condities	
Uitzonderingen	Verbinding die wegvalt met de koersbron. Hier is niets aan te doen en is dus onmacht.

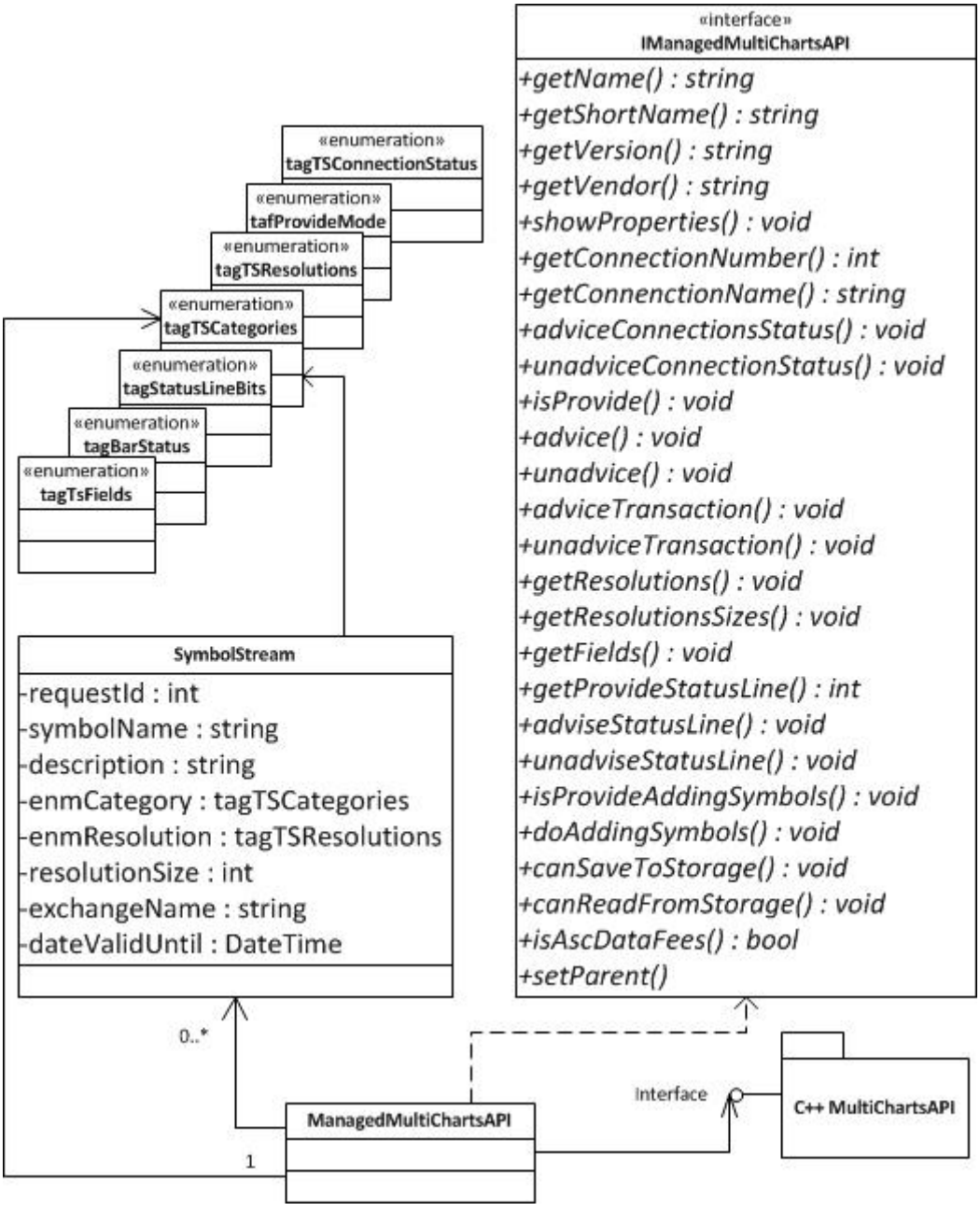
4.2.2 Sequencediagram



Zoals te zien is, maken de classes in het project gebruik van een methode die de Advise methode implementeert van de MultiCharts API. Het begint vanuit MultiCharts, die via de API naar de CVDMSupplier contact maakt met de ManagedDLL. De ManagedDLL spreekt vervolgens het CsharpPart aan en die gaat via een koppeling de BrokerApplicatie vragen om een symbool. De BrokerApplicatie geeft vervolgens door middel van events de data terug aan het CsharpPart via de koppeling en gaat helemaal terug naar MultiCharts.

5. CsharpPart

5.1 Klassendiagram



Dit gedeelte van het gehele project bestaat uit Csharp. Het vertaalt de API naar een Csharp interface. Dit zorgt ervoor dat het managed is.

De IManagedMultiChartsAPI implementeert alle methodes van de ItsSupplier. Verder heeft het nog een extra methode, namelijk setParent. Deze methode is om call backs te kunnen doen naar het C++ gedeelte van de MultiCharts API.

De ManagedMultiChartsAPI implementeert de interface IManagedMultiChartsAPI.

SymbolStream is een class waarin alleen datatypen worden bij gehouden. Het heeft dus geen methodes. SymbolStream is bedacht om een verbinding van koersdata bij te houden met de bijbehorende informatie.

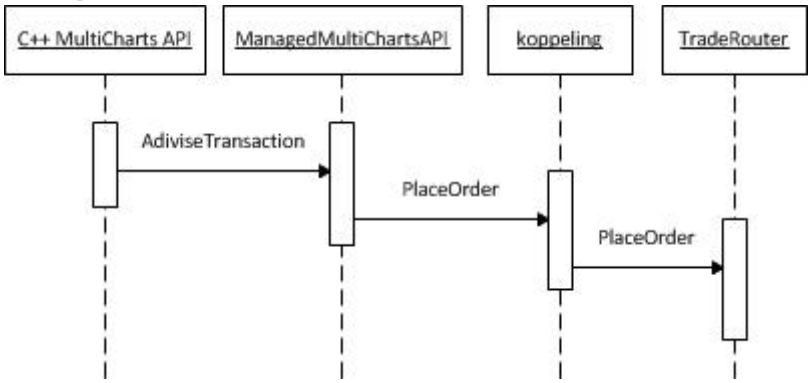
Verder zijn de enums die in de MultiCharts API zijn gedefinieerd nog een keer in het CsharpPart gedefinieerd. Dit wordt gedaan zodat dezelfde waardes in beide lagen worden gehanteerd. De ManagedMultiChartsAPI class maakt van alle enums om de juiste waarden door te geven. De SymbolStream heeft twee variabelen die ieder één van de enums gebruiken.

5.2 Voorbeeld

5.2.1 Use Case

Naam	Plaats order
Primair actor	MultiCharts
Pre-Condities	TradeRouter aan hebben staan MultiCharts gekoppeld hebben met TradeRouter
Hoofd Succes Scenario	1. MultiCharts genereerd signaal 2. De API vangt het signaal op en stuur het door 3. Signaal gaat door de lagen heen 4. Signaal wordt opgepikt door TradeRouter 5. TradeRouter voert order uit die in het signaal zat.
Post-Condities	Signaal is uitgevoerd in vorm van een order
Uitzonderingen	

5.2.2 Sequencediagram



Dit is een voorbeeld over de AdviseTransaction methode van de MultiCharts API. Deze wordt aangeroepen zodra MultiCharts een handelsadvies genereerd. Deze wordt doorgegeven aan de ManagedMultiChartsAPI van het CsharpPart. Die geeft het weer door aan de koppeling die het uiteindelijk weer doorgeeft aan TradeRouter. Zo wordt er een order geplaatst.

5.2.3 Source code

Om het allemaal wat duidelijker te maken volgen een paar stukken code van de API in C++ en de API in C#. Tenslotte is er nog een stuk code die het gebruik van de C# interface duidelijker moet maken.

C++ versie van isProvide

```
…
int IsProvide(long Mode, long Symbol, long Category, long ResolutionSize, long Res
olution, long Field)
        {
                VARIANT_BOOL lngRes = false;
                InitInterface();
                pIMC->isProvide(Mode, Symbol, Category,
                        ResolutionSize, Resolution, Field, lngRes);
                return (int) lngRes;
        }
…
```

Maakt dus gebruik van isProvide van de interface van het C# gedeelte.

C# versie van isProvide

```
…
public void isProvide(int mode, int symbol, int category, int resolutionSize, int
resolution, int field, bool pVal)
        {
            bool blnSuppField = (field == (int)tagTSFIELDS.TS_ASK_FIELD || field =
= (int)tagTSFIELDS.TS_BID_FIELD || field == (int)tagTSFIELDS.TS_TRADE_FIELD);

            bool blnSuppRTMode = (mode == (int)tagProvideMode.RealTimeDataProvide
&& resolution == (int)tagRESOLUTIONS.TS_RESOLUTION_TICK && resolutionSize == 1);

            bool blnSuppHTMode = (mode == (int)tagProvideMode.HistoryDataProvide &
& (resolution == (int)tagRESOLUTIONS.TS_RESOLUTION_TICK && resolutionSize == 1) ||
                                                                                (re
solution == (int)tagRESOLUTIONS.TS_RESOLUTION_MINUTE && resolutionSize == 1));

            pVal = (blnSuppField && (blnSuppRTMode || blnSuppHTMode) ? true : fals
e);
        }
…
```

Gebruik van C# interface van de API

```
…
private IManagedMultiChartsAPI managedAPI;

        public FormCsharp()
        {
            managedAPI = new ManagedMultiChartsAPI();

            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            managedAPI.isProvide(1, 6, 8, 32, 46, 7, true);
        }
…
```

6. Koppeling TradeRouter met MultiCharts API C#

6.1 Klassendiagram

In de koppeling is er alleen maar een class nodig die gebruik maakt van de interface van het CsharpPart en van de interface van de TradeRouter. De Koppelaar class is een soort van doorgeefluik.

Bijlage V

Een alternatief technische analysepakket

Een alternatief technische analysepakket

MetaTrader of NinjaTrader?

Omdat MultiCharts komt te vervallen, moet er verder worden gekeken. Een andere technische analysepakket dus. Er zijn er al twee uitgekozen; MetaTrader en NinjaTrader. Uiteindelijk blijft er één over voor het project.

16-4-2012

Inhoudsopgave

1. Inleiding

De MultiCharts API waarmee eerder gewerkt werd was verouderd en werkte niet met de laatste versie van MultiCharts. Hierdoor werd er bij MultiCharts zelf gevraagd naar een nieuwere API. MultiCharts antwoorde terug met dat het mogelijk is om een licentie af te nemen tegen zeer hoge bedragen. Deze bedragen zijn zo danig hoog, dat de investering niet terug kan worden verdiend. Hierdoor valt MultiCharts af.

Dit valt weer onder het diepteonderzoek. Na dit diepteonderzoek zal er een nieuw oplossingsplan worden gemaakt.

Omdat het project niet perse met MultiCharts gecombineerd dient te worden is, is het mogelijk om een andere Technische Analysepakket te gaan gebruiken. Er zijn er twee voor gesteld door de begeleider; MetaTrader en NinjaTrader.

Het is van belang dat het weer om een universele data adapter gaat. Het zal dus gemakkelijk zijn om een brokerapplicatie te koppelen met één van de technische analysepakketten.

Er komt een onderzoek naar beide pakketten. Er wordt een afweging gemaakt tussen de volgende punten:

- Beschikbaarheid: Is het pakket gemakkelijk te verkrijgen? Zijn er kosten verbonden aan het pakket? Hoe zit het met de support van het bedrijf?
- API: Is de API in een moderne programmeertaal geschreven? Zijn er kosten voor de API?

Er volgt nog een conclusie met punten waarin duidelijk wordt gemaakt of het pakket de voorkeur heeft. Dit document moet leiden naar een keuze tussen MetaTrader en NinjaTrader.

2. MetaTrader 5

2.1 Beschikbaarheid

MetaTrader 5 is vrij te downloaden en te installeren. Wel dien je een account aan te maken waarmee je inlogt in de applicatie. Mocht je MetaTrader buiten de demo omgeving willen gebruiken, dan dien je een betaalde account aan te schaffen.

De support van MetaTrader wil helaas niet goed meewerken. Ze geven niet graag informatie vrij en reageren traag. De informatie die ze dan geven is niet direct een antwoord op mijn vraag en ze geven geen voorbeelden.

2.2 API

Er is geen officiële API, maar wel API's die door onafhankelijke ontwikkelaars zijn geschreven. Deze API's zijn helaas geschreven voor MetaTrader 4. Een voorbeeld hiervan is http://www.mt4api.net/.

Verder kosten die API's ook geldt, terwijl er dan geen garantie is dat het werkt, of dat het alle functionaliteit heeft die je wenst.

2.3 Conclusie

Het zou veel onderzoek tijd kosten om een eigen API te gaan bouwen. Dit omdat er kennis is van de architectuur van MetaTrader en de functionaliteit ervan.

Tevens is de support van MetaTrader niet erg behulpzaam, waardoor vragen niet meteen beantwoord worden, of zij een vraag terug stellen. Dit zal nog meer tijd kosten en maakt het zeer onaantrekkelijk om een koppeling te bouwen tussen TradeRouter en MetraTrader.

3. NinjaTrader 7

3.1 Beschikbaarheid

NinjaTrader 7 is vrij te downloaden en te installeren. Wel wordt er van je gevraagd om een account aan te maken om te kunnen inloggen in de applicatie. Met het account kun je de applicatie testen in een demo-omgeving.

Verder is het mogelijk om verbindingen te maken met brokerapplicaties, bijv Trader Workstation, om echte data binnen te krijgen. Met deze data is het mogelijk om strategieën toe te passen. De orders die worden uitgevoerd blijven echter wel in NinjaTrader zelf en zie je dus niet terug bij Trader Workstation.

De support van NinjaTrader is heel erg behulpzaam. Ze reageren snel op vragen en geven volledige antwoorden.

Mocht je er niet uitkomen met de support, dan is er ook nog een forum te vinden op de website van NinjaTrader, waarbij ontwikkelaars elkaar proberen te helpen.

3.2 API

NinjaTrader heeft een API die geleverd wordt met de applicatie. Deze API bestaat uit een DLL bestand genaamd: NinjaTrader.Client.dll. Deze is er zowel in de 32- als de 64-bit variant. Het is geschreven in .Net en dus gemakkelijk te gebruiken in C#.

Er is ook veel documentatie te vinden op: http://www.ninjatrader.com/support/helpGuides/nt7/index.html?automated_trading_interface_at.htm. Dit maakt het gebruik heel gemakkelijk om de API van NinjaTrader te gebruiken.

Echter kun je met de API alleen koersdata naar NinjaTrader versturen. Je kunt geen signalen van NinjaTrader ontvangen via de API.

3.3 Conclusie

Een groot pluspunt van het gebruiken van NinjaTrader is dat er een API beschikbaar in de .NET omgeving. Hierdoor is het gemakkelijk te importeren en aanroepen te doen.

Tevens is er duidelijke informatie te vinden en werkt de support van NinjaTrader prettig mee. Dit allemaal zorgt ervoor dat weinig onderzoek en dus weinig tijd nodig is om een koppeling tussen TradeRouter en NinjaTrader te realiseren.

Een klein nadeel is dat omdat de API van NinjaTrader geen order signalen terug sturen. Hiervoor moet er een omweg worden ontworpen.

4. Conclusie

Aangezien MetaTrader geen API heeft, is keuze gevallen voor NinjaTrader. Het heeft immers geen zin om een koppeling te maken voor een pakket dat geen API heeft.

Bijlage VI

Requirements FrontEnd

FrontEnd Requirements

Eisen voor de software

Hoe de verschillende onderdelen van het project er komen uit te zien. Ze moeten aan bepaalde eisen voldoen en volgens een bepaalde architectuur gebouwd worden.

23-4-2012

Inhoudsopgave

1. Inleiding

Na het tweede diepte onderzoek, is het nu mogelijk om het nieuwe oplossingsplan te ontwikkelen.

Het koppelen van TradeRouter met NinjaTrader gebeurd door middel van het maken van lagen in de software en die met elkaar te verbinden. Via deze lagen kan TradeRouter koersdata leveren aan NinjaTrader. NinjaTrader analyseert deze aangeleverde koersdata en genereert handelsadviezen. Deze handelsadviezen worden weer gestuurd naar TradeRouter die er dan weer mee automatisch gaat handelen.

Dit is in grote lijnen de opzet van de software. Met deze opzet wordt er rekening gehouden met:

- Wijzigingen
- Onderhoud
- Vernieuwingen

Er zijn drie lagen aanwezig in het project:

- NinjaTrader Client API
- NinjaTrader FrontEnd Application
- NinjaTrader Trading DLL

Per laag wordt er uitgelegd wat zijn rol is in het geheel, wat de kenmerken zijn en hoe het gebruikt wordt.

Ook zijn er nog bepaalde eisen waaraan moeten worden voldaan vanuit de opdrachtgever. Deze eisen moeten worden verwerkt in het project om het slagen te verzekeren.

Om het implementatiegedeelte van het project zo soepel mogelijk te laten verlopen, moet er eerst goed voorbereid worden. Het helpt om de lagen nader te bestuderen en goed uit te schrijven, zodat het duidelijk wordt hoe deze moeten worden geïmplementeerd.

2. Gelaagde structuur

2.1 Voordelen

Tegenwoordig wordt software vaak in lagen geschreven. Deze vorm van software architectuur wordt ook wel Multilagenarchitectuur genoemd. Een veel voorkomende vorm van deze architectuur is een 3-lagenachitectuur. Hierin is er een afzonderlijke laag voor de presentatie, business logica en datatoegang/dataopslag. Het is mogelijk om nog meer lagen in de architectuur te werken, om zo iedere laag een specifieke rol te geven. Deze architectuur heeft vele voordelen te bieden aan programmeurs.

Doordat de software is opgedeeld in lagen, is het ook weer mogelijk om functionaliteit te ontkoppelen en te vervangen door een andere laag in te zetten. De lagen worden aan elkaar gekoppeld en werken zo als een geheel.

Wat er ook nog bij komt kijken is dat wanneer er een wijziging moet plaatsvinden in een bepaalde laag, dat de andere lagen hier niets van merken. Zo is het gemakkelijk om per laag te werken, zonder de andere lagen te beïnvloeden. Dit maakt het opsporen van fouten ook eenvoudiger, omdat de programmeur weet in welke laag hij als laatste aanpassingen heeft verricht.

Tevens kan een laag worden hergebruikt in een ander systeem, waardoor er niets hoeft worden te herschreven. Een laag heeft dus zijn eigen verantwoordelijkheden en kan onafhankelijk ingezet worden in meerdere systemen.

Als laatste, maar niet het minste, is een zeer groot voordeel dat er beter parallel gewerkt kan worden. Functionaliteit kan immers opgedeeld worden in de verschillende lagen. Omdat iedere laag afzonderlijk kan werken, hoeven de programmeurs ook niet op elkaar te wachten om te testen. Dit verhoogt de productiviteit en ook kwaliteit van het werk, omdat een programmeur goed bekend is met de laag waarin hij heeft gewerkt.

2.2 In het project

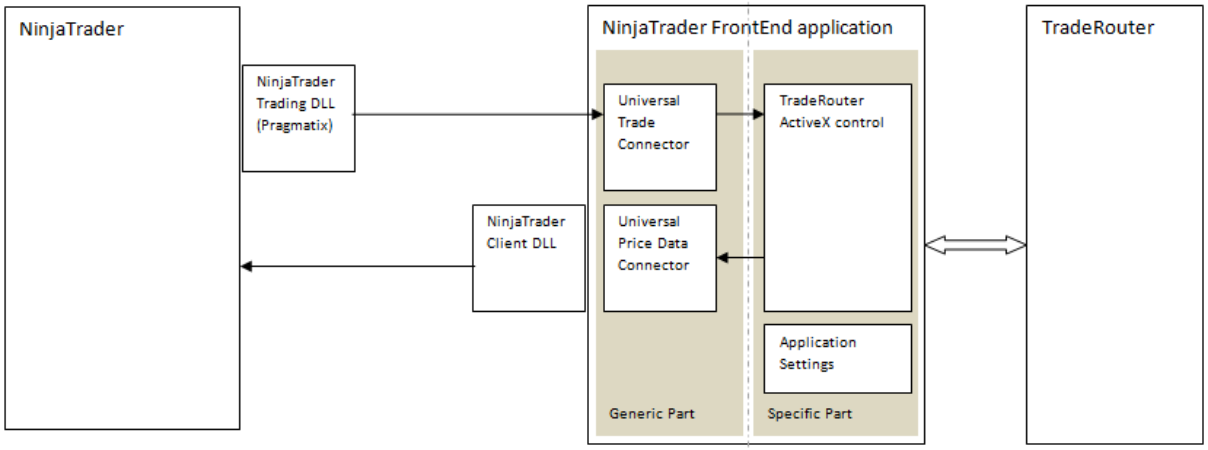
In het project wordt er onderscheid gemaakt in drie lagen:

- NinjaTrader Client API
- NinjaTrader FrontEnd Application
- NinjaTrader Trading DLL

De NinjaTrader Client API wordt aangeleverd door NinjaTrader Inc. De API wordt als een DLL aangeleverd die je in een C# project als een assembly kunt gebruiken.

De NinjaTrader API kun je dus in je eigen project gebruiken en aanroepen. Je kunt de methodes aanroepen om data te versturen en op te vragen aan de DLL.

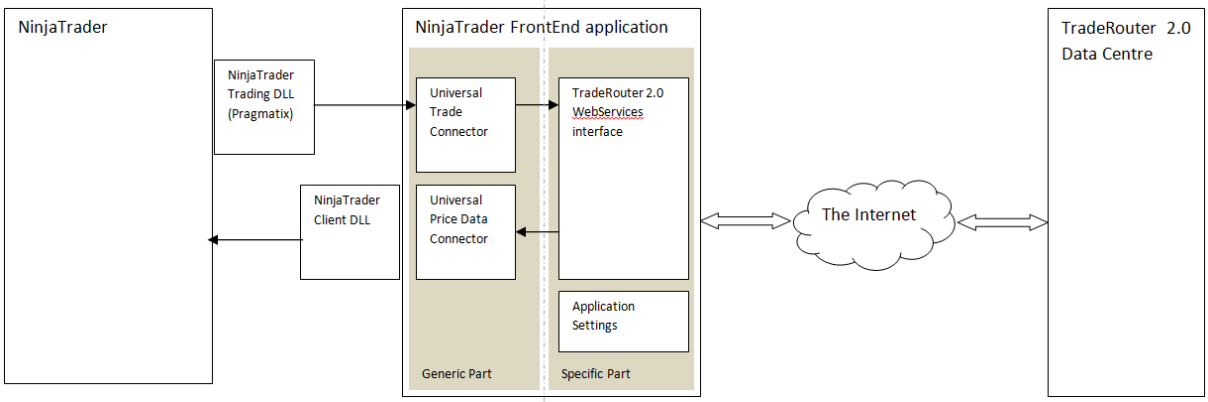
Omdat de API helaas niet terugkoppelt als er een signaal is gegenereerd, is er nog een DLL nodig. Dit is de NinjaTrader Trading DLL. Deze zal ontwikkeld worden door de afstudeerder. De bedoeling is dat deze DLL gebruikt wordt in een NinjaScript. De DLL wordt dan geïmporteerd. In deze DLL staan methodes die aanroepen doen naar de NinjaTrader FrontEnd applicatie.



Zoals te zien is aan het diagram, zal de communicatie, tussen NinjaTrader en de FrontEnd, via de DLL's lopen. Dit zorgt ervoor dat de code om te communiceren gescheiden is van de business logica.

Verder laat het diagram zien, dat de NinjaTrader FrontEnd direct kan communiceren met de TradeRouter. Dit komt omdat er een ActiveX component in de FrontEnd zit die kan communiceren met de TradeRouter. In het FrontEnd gedeelte van het document wordt hier meer aandacht aan besteed.

Tenslotte is er een gedeelte in de FrontEnd generiek, een gedeelte specifiek. Het generieke gedeelte zal maar één keer ontwikkeld hoeven te worden voor aan de kant van NinjaTrader. Het specifieke gedeelte zou voor iedere brokerapplicatie geschreven moeten worden. Dit zorgt ervoor dat de FrontEnd voor meerdere brokerapplicaties gebruikt kan worden.



3. Requirements

De opdrachtgever heeft bepaalde eisen opgesteld. Om het project goed te laten beoordelen, moet er worden voldaan aan de deze eisen. Als deze er niet in verwerkt zitten, dan zal de opdracht gever niet tevreden zijn met het product en het project met een onvoldoende te beoordelen.

3.1 Niet-functionele eisen

Betrouwbaarheid. Hiermee wilt de opdrachtgever zeggen dat de lagen altijd verbindingen met elkaar moeten hebben en met de applicatie waarmee ze werken. Deze eis komt voort uit dat koersdata binnen een seconde alweer verouderd is. Mocht er enkele seconden geen verbinding zijn, dan mist de gebruiker al waardevolle informatie. Hierdoor kan de gebruiker zonder het te weten handelen met verouderde koersdata en is dat nadelig voor de gebruiker. Hierdoor zal de gebruiker de applicatie als onbetrouwbaar beschouwen.

Efficiënt. De applicatie moet efficiënt om kunnen gaan met de koersdata die binnenkomt. De koersdata kan mogelijk meerdere malen per seconde binnenkomen. De gebruiker zal waarschijnlijk meerdere koersen volgen. Dit houdt in dat er vele koersdata tegelijkertijd binnenkomt en dat deze allemaal moet worden doorgestuurd. Het is dus van belang dat al deze data snel verwerkt wordt, zonder dat er data kwijt geraakt wordt. Anders komt de betrouwbaarheid in gevaar.

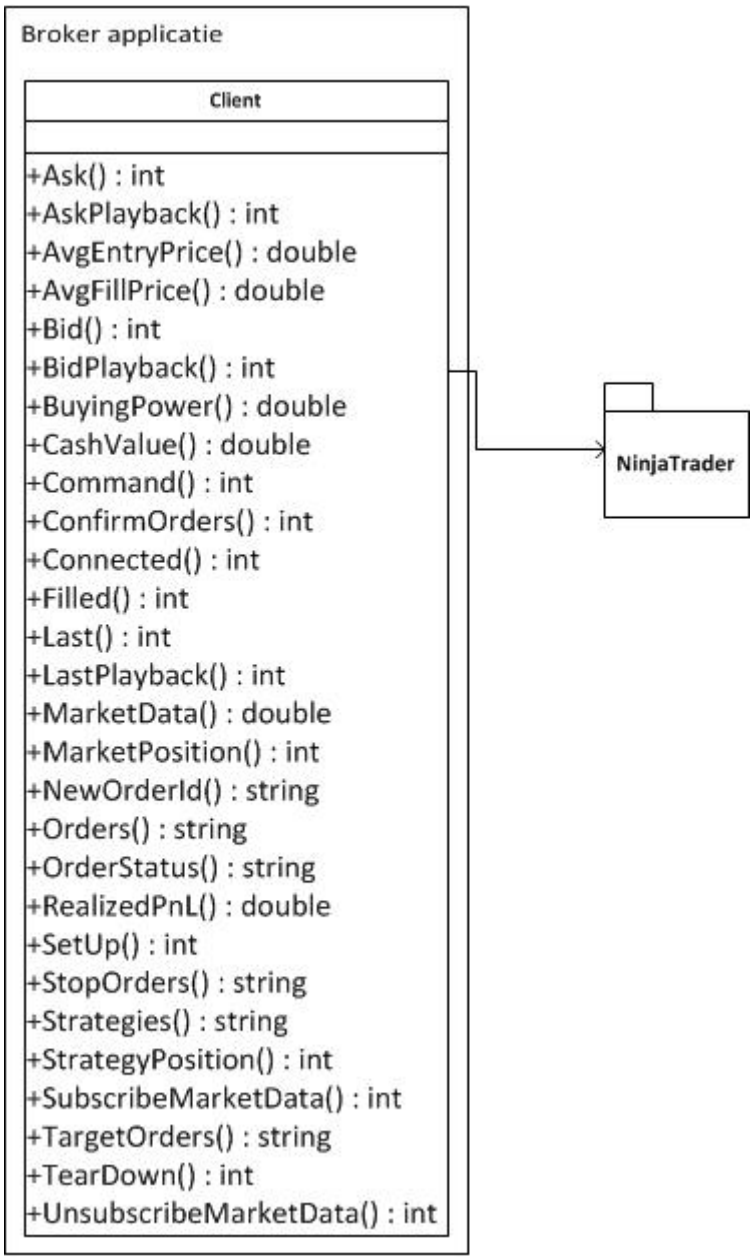
Flexibel inzetbaar. Dit wordt door middel van de multilagenarchitectuur verwezenlijkt. De verschillende lagen maakt het mogelijk, door een andere bron van koersinformatie gemakkelijk te kunnen koppelen. Het zal uiteindelijk gekoppeld worden aan de bovenste laag.

3.2 Functionele eisen

De communicatie tussen vanaf de NinjaTrader Trading DLL moet via sockets verlopen. Dit heeft vooral te maken met het feit dat in de toekomst TradeRouter vervangen gaat worden. De vervanging van TradeRouter wordt een web-based applicatie.

4. NinjaTrader Client DLL

4.1 Klassendiagram



Zoals te zien is, is het een vrij simpel klassendiagram. Dit komt omdat de kern van de NinjaTrader API aangeleverd komt vanuit NinjaTrader Inc. Deze DLL wordt verder beschreven in het “NinjaTrade.Client DLL” document dat als bijlage is toegevoegd.

De class Client heeft alle methodes die gebruikt worden in de NinjaTrader API. In de methodes wordt de data die vanuit de brokerapplicatie komt doorgegeven aan NinjaTrader.

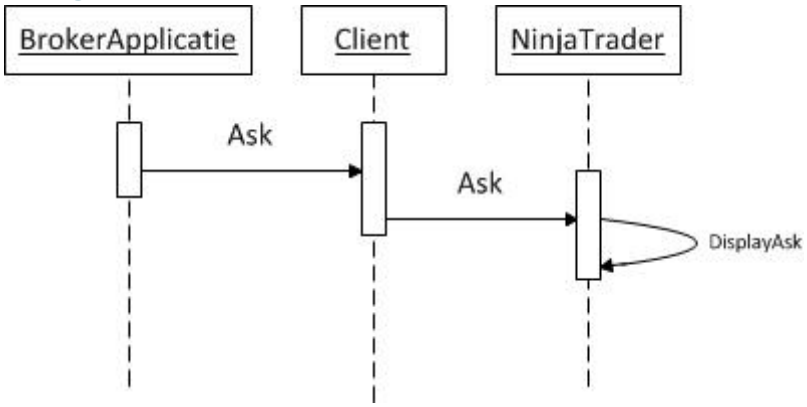
De brokerapplicatie heeft dus een instantie van een Client binnen haar omgeving. Zo kan er vanuit een eigen applicatie data worden verstuurd naar NinjaTrader. Met deze opzet is het alleen mogelijk om koersdata door te geven aan NinjaTrader. Het is dus niet mogelijk om handelssignalen op te vangen van NinjaTrader.

4.2 Voorbeeld

4.2.1 Use Case

Naam	Stuur Ask informatie door
Primair actor	Gebruiker
Pre-Condities	TradeRouter en NinjaTrader draaiende hebben
Hoofd Succes Scenario	4. Gebruiker opent de FrontEnd applicatie 5. Gebruiker vraagt de beschikbare engines op 6. Wordt meteen door gestuurd
Post-Condities	NinjaTrader laat de beschikbare engines zijn in de het instrument menu. De Data van de engines worden real time doorgestuurd naar NinjaTrader.
Uitzonderingen	Verbinding die wegvalt met de koersbron. Hier is niets aan te doen en is dus onmacht.

4.2.2 Sequencediagram



De Broker applicatie roept de Ask methode aan van de Client. De Broker geeft de parameters mee die moeten worden weergeven in NinjaTrader. Zodra NinjaTrader deze binnen krijgt van de Client, wordt het weergeven in NinjaTrader.

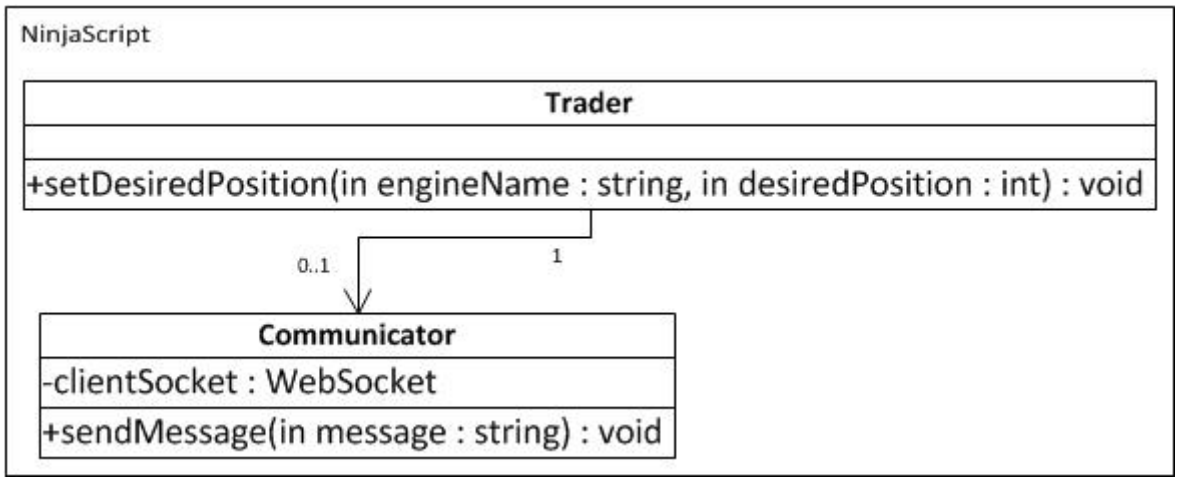
4.2.3 Source code

```
using NinjaTrader.Client;

private Client client = new Client();

private void AxctrTrMcFrontEnd1OnPrice(object sender, __TRFrontEnd_PriceEvent tRPr
iceEvent)
  {
      client.Ask(ctrTrMcFrontEndPriceEvent.engine, tRPriceEvent.ask,
        ctrTrMcFrontEndPriceEvent.askSize);

      client.Bid(ctrTrMcFrontEndPriceEvent.engine, tRPriceEvent.bid,
        ctrTrMcFrontEndPriceEvent.bidSize);

      client.Last(ctrTrMcFrontEndPriceEvent.engine, tRPriceEvent.price,
        ctrTrMcFrontEndPriceEvent.lastTradeSize);
  }
```

De code is een simpel voorbeeld hoe de DLL gebruikt kan worden. In dit voorbeeld wordt er, zodra een prijs binnenkomt, die doorgeven aan de Client van NinjaTrader. Zo wordt de data real-time doorgegeven.

5. NinjaTrader Trading DLL

5.1 Klassendiagram



De Trading DLL bestaat uit tweetal classes. Trader is de hoofdclass waar vanuit een NInjaScript tegen gesproken wordt. Aan de Trader wordt gevraagd om een bepaalde positie in te nemen op een bepaald financiële product.

De Communicator stuurt de orders door naar de FrontEnd applicatie. De Communicator krijgt deze orders van de Trader. De Communicator verwerkt de orders naar een bepaald formaat die door de FrontEnd geïnterpreteerd kan worden.

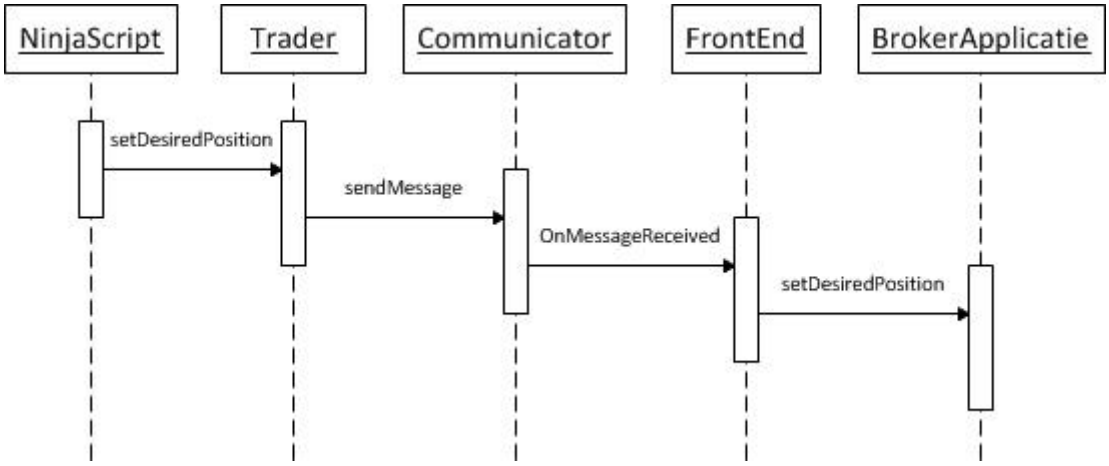
Er zit een scheiding tussen de communicatie en de het binnen krijgen van signalen. Dit is zo gemaakt voor leesbaarheid van de code.

5.2 Voorbeeld

5.2.1 Use Case

Naam	Neem positie in
Primair actor	NinjaScript
Pre-Condities	TradeRouter aan hebben staan NinjaTrader gekoppeld hebben met TradeRouter
Hoofd Succes Scenario	6. NinjaScript genereerd signaal 7. NinjaScript geeft order aan Trader 8. Trader wordt doorgegeven aan Communicator 9. Communicator zet het om in een begrijpbaar formaat 10. Bericht doorsturen
Post-Condities	Signaal is uitgevoerd in vorm van een order
Uitzonderingen	Verbinding die wegvalt met de Broker. Dit is ook al onmacht

5.2.2 Sequencediagram



Dit is een voorbeeld over de setDesiredPosition methode van de ActiveX control van TradeRouter. Deze wordt aangeroepen zodra NinjaScript een conditie heeft waar gemaakt. Deze wordt doorgegeven aan de Communicator, die er een bericht, in een bepaald formaat van maakt. Die stuurt het weer door naar de FrontEnd applicatie. De FrontEnd applicatie zorgt er uiteindelijk voor dat het bij de Broker terecht komt.

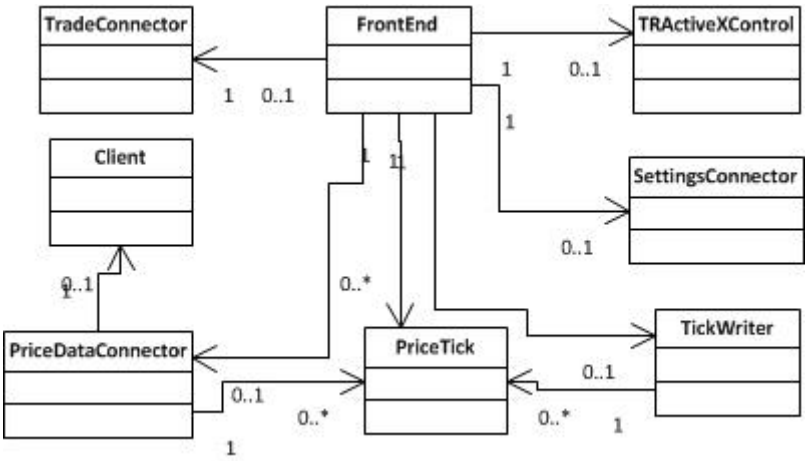
5.2.3 Source code

Hier volgt een voorbeeld van een NinjaScript die gebruik maakt van een geïmporteerde DLL (TradeRouterNinjaTrader). Het is zoals te zien is gewoon te gebruiken in een NinjaScript. Dit komt omdat NinjaScript een extensie is op C# is.

```
#region Using declarations
using System;
using System.ComponentModel;
using System.Diagnostics;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Xml.Serialization;
using NinjaTrader.Cbi;
using NinjaTrader.Data;
using NinjaTrader.Gui.Chart;
using TradeRouterNinjaTrader;
#endregion

// This namespace holds all strategies and is required. Do not change it.
namespace NinjaTrader.Strategy
{
    /// <summary>
    /// Enter the description of your strategy here
    /// </summary>
    [Description("Enter the description of your strategy here")]
    public class TRTest : Strategy
    {
        #region Variables
        // Wizard generated variables
        // User defined variables (add any user defined variables below)
        TradeRouterClient client;
        #endregion

        /// <summary>
        /// This method is used to configure the strategy and is called once befor
e any strategy method is called.
        /// </summary>
        protected override void Initialize()
        {
            CalculateOnBarClose = true;
            client = new TradeRouterClient();
        }

        /// <summary>
        /// Called on each bar update event (incoming tick)
        /// </summary>
        protected override void OnBarUpdate()
        {
            client.sendMessage("Test");
        }
    }
}
```

6. FrontEnd

6.1 Klassendiagram



De FrontEnd is de beheerder van deze laag. De FrontEnd zorgt ervoor dat data verstuurd wordt naar NinjaTrader. Ook zorgt het ervoor dat orders naar de broker, in dit geval TradeRouter, verstuurd worden.

De TradeConnector is het aanspreek punt voor de NinjaScript. Het wacht op een bericht. Zodra er een bericht binnen is, wordt datzelfde bericht, ontcijferd. Via een event wordt de FrontEnd op de hoogte gebracht (door middel van een event) dat het bericht binnen is.

Met het event dat is opgegooid, geeft de FrontEnd een opdracht aan de TRActiveXControl. Dat opdracht houdt een order in. De TRActiveXControl stuurt de order door naar de TradeRouter, die op zijn beurt er voor zorgt dat het uitgevoerd wordt.

Verder gooit de TRActiveXControl events op zodra er prijsdata binnen komt. De FrontEnd maakt een PriceTick object van de binnengekomen data. In de PriceTick staat de koersdata en de naam van de engine. De TickWriter wordt dan aan het werk gezet. De TickWriter zit dan een PriceTick in een dictionary. Als de enginenaam al voorkomt, dan wordt dezelfde PriceTick overschreven met de nieuwe data. Anders wordt de PriceTick toegevoegd aan de dictionary. Bij deze gebeurtenis wordt een event opgegooid.

De PriceDataConnector pikt de event op van de TickWriter. Die haalt dan de PriceTick uit de dictionary. De PriceDataConnector haalt de koersdata uit de PriceTick en geeft die dan door aan de Client. De Client, dat afkomstig is uit de NinjaTrader DLL, stuurt de prijsdata door aan NinjaTrader.

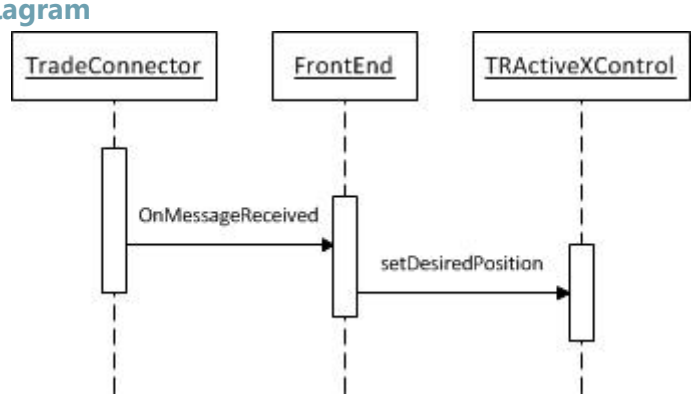
Tenslotte is er nog de SettingsConnector. Deze class zorgt ervoor dat een configuratiebestand kan worden ingeladen en wordt uitgevoerd. Tevens zorgt het ervoor dat er een configuratie gewijzigd en opgeslagen kan worden.

6.2 Voorbeeld

6.2.1 Use Case

Naam	Stuur bericht door
Primair actor	FrontEnd
Pre-Condities	TradeRouter aan hebben staan NinjaTrader gekoppeld hebben met TradeRouter
Hoofd Succes Scenario	1. TradeConnector krijgt een bericht binnen 2. TradeConnector ontcijfert het bericht en gooit een event op 3. FrontEnd reageert op de event en stuurt de order naar TRActiveXControl 4. TRActiveControl verstuurd het naar de TradeRouter
Post-Condities	Signaal is uitgevoerd in vorm van een order
Uitzonderingen	De verbinding met de TradeRouter valt weg. Dit is onmacht.

6.2.2 Sequencediagram



Zoals te zien is, komt er een bericht binnen bij de TradeConnector. Die gooit vervolgens een event op, welk opgepikt wordt door de FrontEnd. De FrontEnd verstuurt de order weer naar de TRActiveXControl.

Bijlage I: NinjaTrader.Client DLL Interface Functions

int Ask(string instrument, double price, int size)
Sets the ask price and size for the specified instrument. A return value of 0 indicates success and -1 indicates an error.

int AskPlayback(string instrument, double price, int size, string timestamp)
Sets the ask price and size for the specified instrument for use when synchronizing NinjaTrader playback with an external application playback. A return value of 0 indicates success and -1 indicates an error. The *timestamp* parameter format is "yyyyMMddhhmmss".

double AvgEntryPrice(string instrument, string account)
Gets the average entry price for the specified instrument/account combination.

double AvgFillPrice(string orderId)
Gets the average entry price for the specified orderId.

int Bid(string instrument, double price, int size)
Sets the bid price and size for the specified instrument. A return value of 0 indicates success and -1 indicates an error.

int BidPlayback(string instrument, double price, int size, string timestamp)
Sets the bid price and size for the specified instrument for use when synchronizing NinjaTrader playback with an external application playback. A return value of 0 indicates success and -1 indicates an error. The *timestamp* parameter format is "yyyyMMddhhmmss".

double BuyingPower(string account)
Gets the buying power for the specified account. *Not all brokerage technologies support this value.

double CashValue(string account)
Gets the cash value for the specified account. *Not all brokerage technologies support this value.

int Command(string command, string account, string instrument, string action, int quantity, string orderType, double limitPrice, double stopPrice, string timeInForce, string oco, string orderId, string strategy, string strategyId)
Function for submitting, cancelling and changing orders, positions and strategies. Refer to the Commands and Valid Parameters section for detailed information. A return value of 0 indicates success and -1 indicates an error. The Log tab will list context sensitive error information.

int ConfirmOrders(int confirm)
The parameter confirm indicates if an order confirmation message will appear. This toggles the global option that can be set manually in the NinjaTrader Control Center by selecting the Tools menu and the menu item Options, then checking the "Confirm order placement" checkbox. A value of 1 sets this option to true, any other value sets this option to false.

int Connected(int showMessage)
Returns a value of zero if the DLL has established a connection to the NinjaTrader server (application) and if the ATI is currently enabled or, -1 if it is disconnected. Calling any function in the DLL will automatically initiate a connection to the server. The parameter *showMessage* indicates if a message box is displayed in case the connection cannot be established. A value of 1 = show message box, any other value = don't show message box.

int Filled(string orderId)
Gets the number of contracts/shares filled for the orderId.

int Last(string instrument, double price, int size)
Sets the last price and size for the specified instrument. A return value of 0 indicates success and -1 indicates an error.

int LastPlayback(string instrument, double price, int size, string timestamp)
Sets the last price and size for the specified instrument for use when synchronizing NinjaTrader playback with an external application playback. A return value of 0 indicates success and -1 indicates an error. The *timestamp* parameter format is "yyyyMMddhhmmss".

double MarketData(string instrument, int type)
Gets the most recent price for the specified instrument and data type. 0 = last, 1 = bid, 2 = ask. You must first call the SubscribeMarketData() function prior to calling this function.

int MarketPosition(string instrument, string account)
Gets the market position for an instrument/account combination. Returns 0 for flat, negative value for short positive value for long.

string NewOrderId()
Gets a new unique order ID value.

string Orders(string account)
Gets a string of order ID's of all orders of an account separated by '|'. *If a user defined order ID was not originally provided, the internal token ID value is used since it is guaranteed to be unique.

string OrderStatus(string orderId)
Gets the order state (see definitions) for the orderId. Returns an empty string if the order ID value provided does not return an order.

double RealizedPnL(string account)
Gets the realized profit and loss of an account.

int SetUp(string host, int port)
Optional function to set the host and port number. By default, host is set to "localhost" and port is set to 36973. The default port number can be set via the General tab under Options. If you change these default values, this function must be called before any other function. A return value of 0 indicates success and -1 indicates an error.

string StopOrders(string strategyId)
Gets a string of order ID's of all Stop Loss orders of an ATM Strategy separated by '|'. Internal token ID value is used since it is guaranteed to be unique.

string Strategies(string account)
Gets a string of strategy ID's of all ATM Strategies of an account separated by '|'. Duplicate ID values can be returned if strategies were initiated outside of the ATI.

int StrategyPosition(string strategyId)
Gets the position for a strategy. Returns 0 for flat, negative value for short and positive value for long.
int SubscribeMarketData(string instrument)

Starts a market data stream for the specific instrument. Call the MarketData() function to retrieve prices. Make sure you call the UnSubscribeMarketData() function to close the data stream. A return value of 0 indicates success and -1 indicates an error.

string TargetOrders(string strategyId)
Gets a string of order ID's of all Profit Target orders of an ATM Strategy separated by '|'. Internal token ID value is used since it is guaranteed to be unique.

int TearDown()
Disconnects the DLL from the NinjaTrader server. A return value of 0 indicates success and -1 indicates an error.

int UnsubscribeMarketData(string instrument)
Stops a market data stream for the specific instrument. A return value of 0 indicates success and -1 indicates an error.

Bron:
http://www.ninjatrader.com/support/helpGuides/nt7/index.html?automated_trading_interface_at.htm

Bijlage VII

FrontEnd Manual

Manual for the FrontEnd

How to connect TradeRouter with NinjaTrader

15-5-2012

Table of Contents

1. Introduction

NinjaTrader has a lot of connections with brokers, but not all of them. It's not possible to use all available exchange data in NinjaTrader. The link between NinjaTrader and TradeRouter will help to eliminate this limitation. All exchange data that TradeRouter receives, can be send to NinjaTrader. This way, more exchange data can be used in NinjaTrader.

A few steps have to be made to make this link possible. These steps are divided in three categories:

- TradeRouterTrading DLL
- NinjaScript
- FrontEnd application

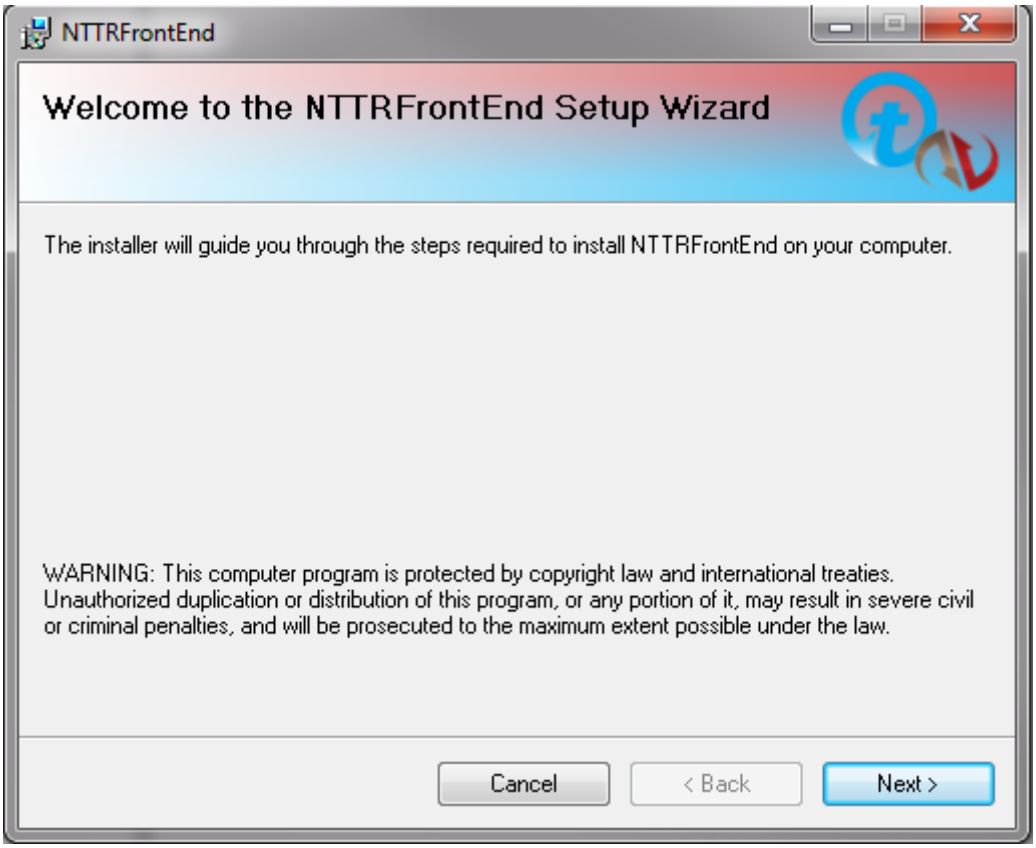
The following software is necessary to make the link successful.

- TradeRouter Lite version 2.1.12 or higher
- NinjaTrader 7 or higher
- TradeRouterTrading DLL
- FrontEnd application version 1.0.0.0 or higher

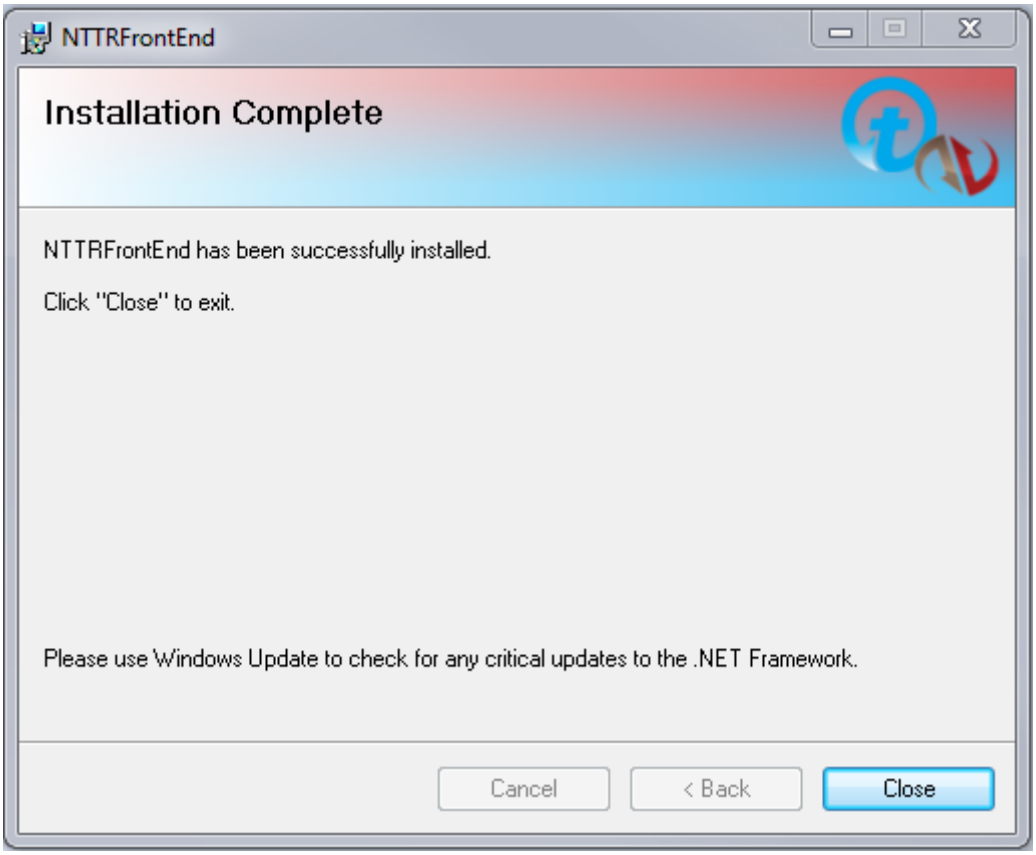
2. FrontEnd application part I

2.1 Installation

Open the NTTRFrontEnd.msi by double left-clicking on the file. You will see the next window.

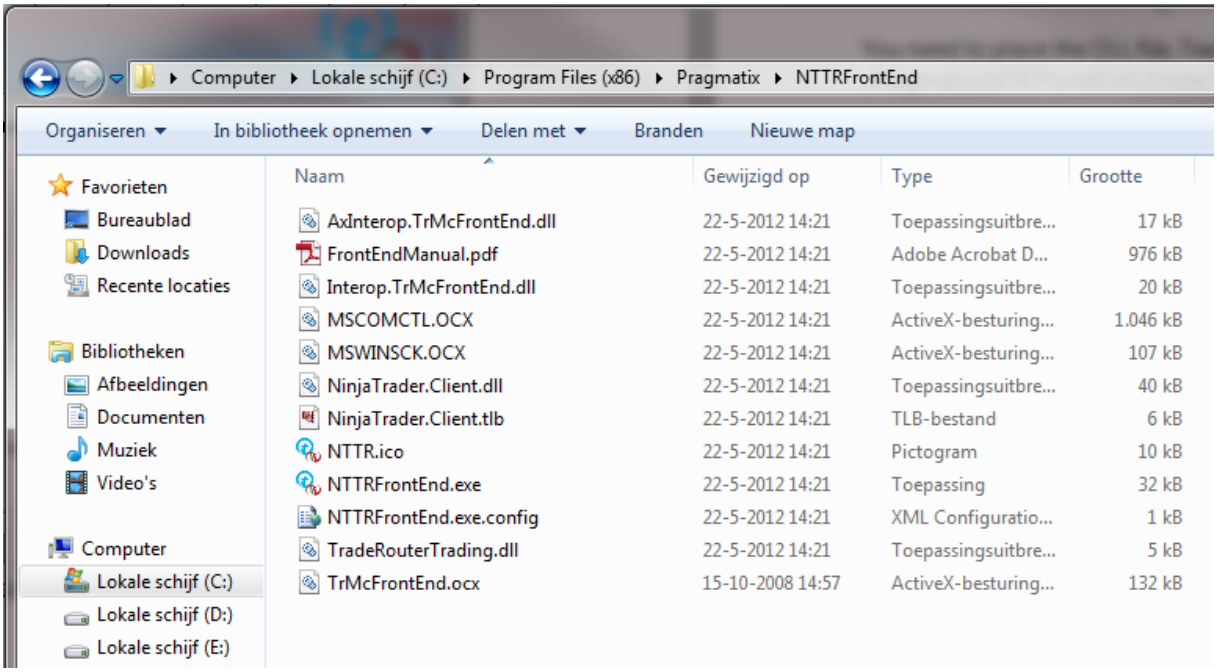


Keep clicking on next until you see this window.



Click now on 'Close'.

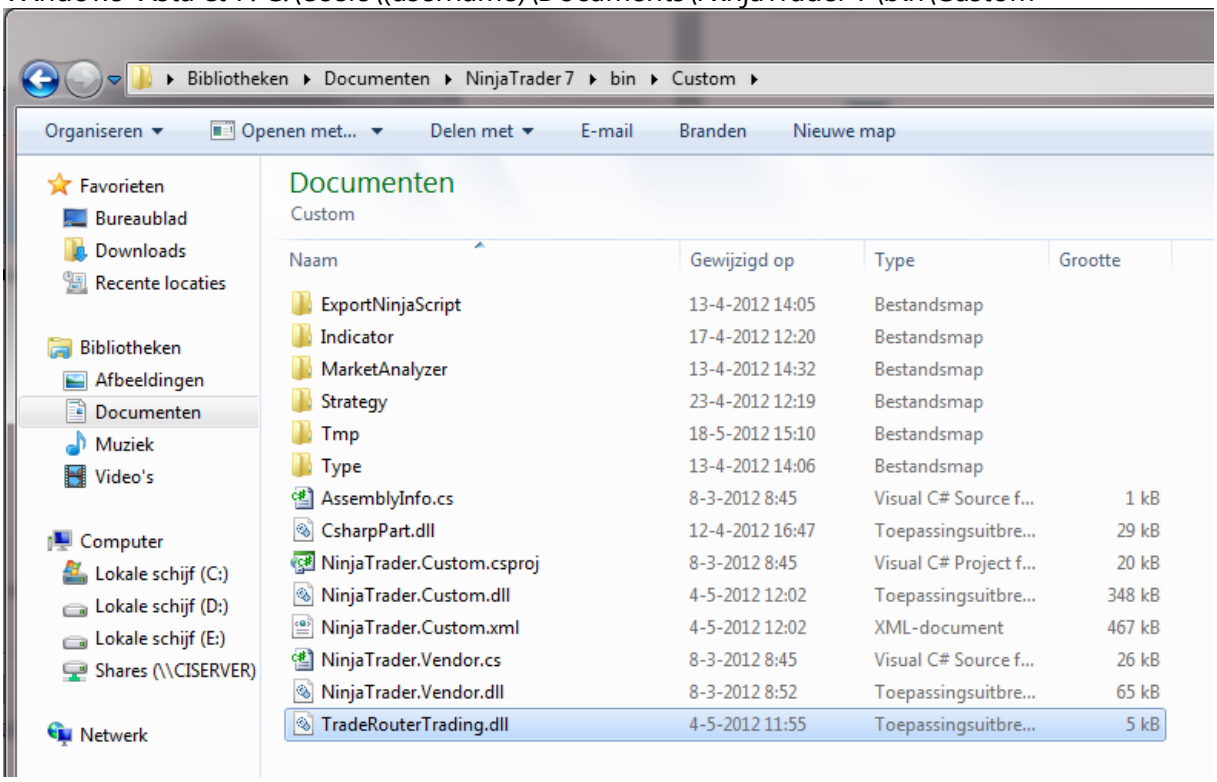
The application can now be found at 'C:\Program Files (x86)\Pragmatix\NTTRFrontEnd'. If you're running a 32bit Windows the path will be C:\Program Files\Pragmatix\NTTRFrontEnd.



3. TradeRouterTrading.dll

You need to place the DLL file, TradeRouterTrading.dll, in the following folder:

Windows XP: C:\Documents and Settings\(username)\Documents\NinjaTrader 7\bin\Custom
Windows Vista & 7: C:\Users\(username)\Documents\NinjaTrader 7\bin\Custom



Make sure NinjaTrader is not running while placing the DLL file. NinjaTrader loads all the DLL files at start-up.

3.1 DLL Contents

Trader Class

Constructors

Trader()
Use this constructor when the FrontEnd is running on the same computer.

Trader(int port)
Use this constructor when FrontEnd is running on the same computer but with another port than the default (3333) port.

Trader(string host, int port)
Use this constructor when FrontEnd is running on another computer.

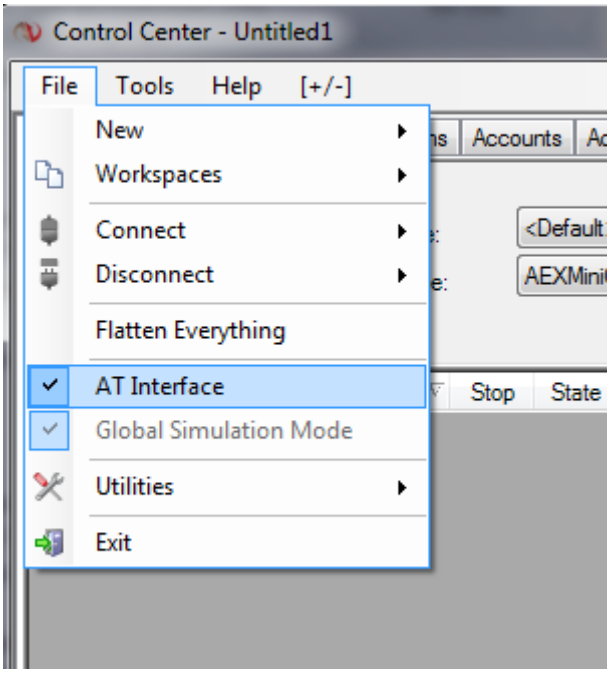
Functions

void setDesiredPosition(string engineName, int DesiredPosition)
With this function you can set the desired position, for an engine, in TradeRouter.

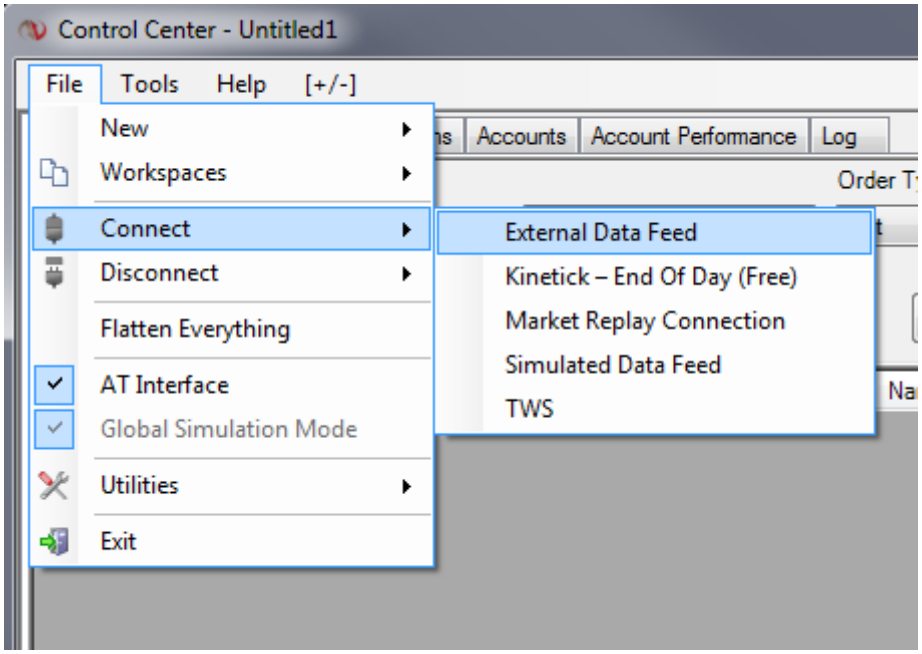
4. FrontEnd application part II

4.1 Connecting with TradeRouter and NinjaTrader

Start NinjaTrader. When NinjaTrader is up and running you need to turn on the Automated Trading Interface (ATI). This is done by left-clicking on 'File->AT Interface" You will see a check sign meaning it is on.



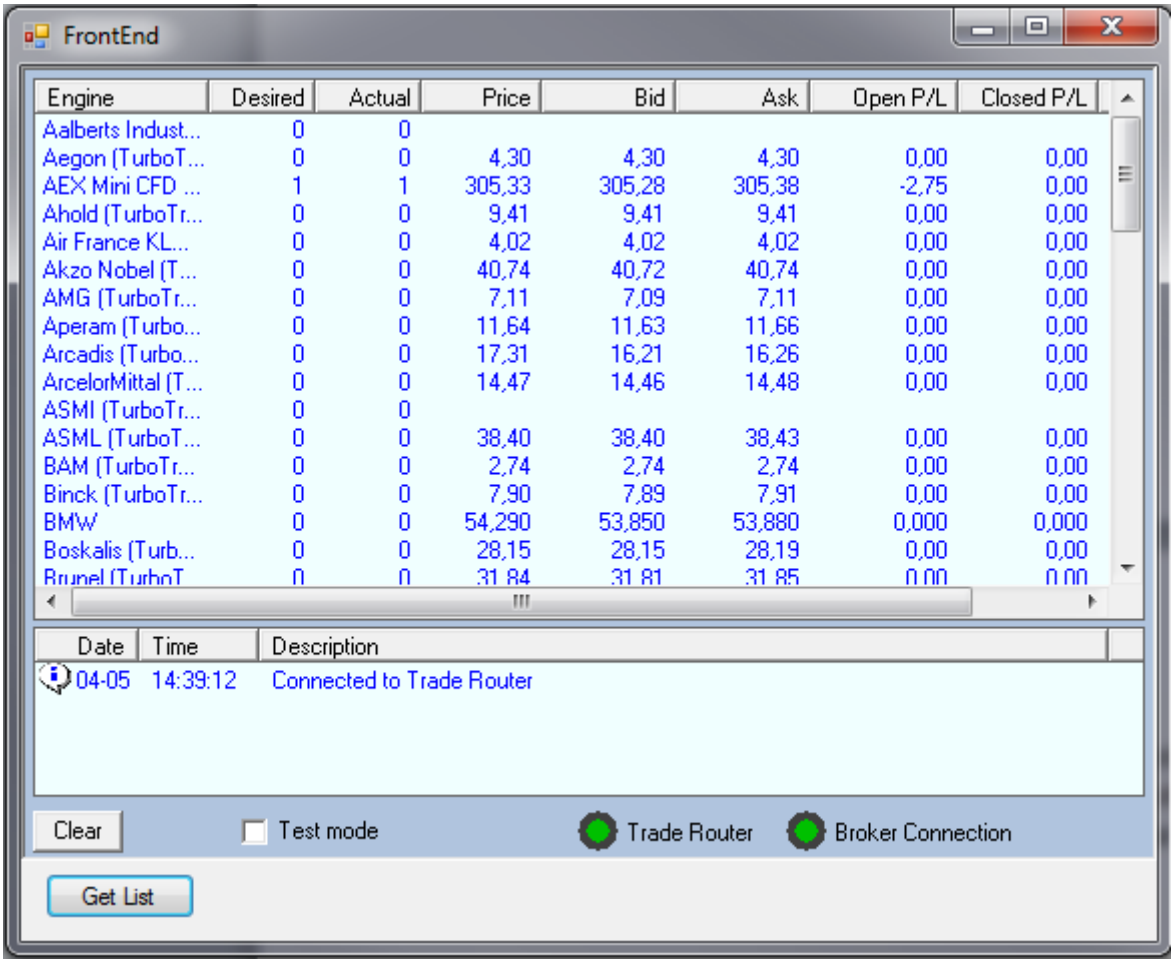
With the ATI turned on, it will accept incoming connection through the ATI. To finalise the connection, you will need to connect to an external data feed. Left-Click on 'File->Connect->External Data Feed'.



NinjaTrader will now wait for the FrontEnd's price data.

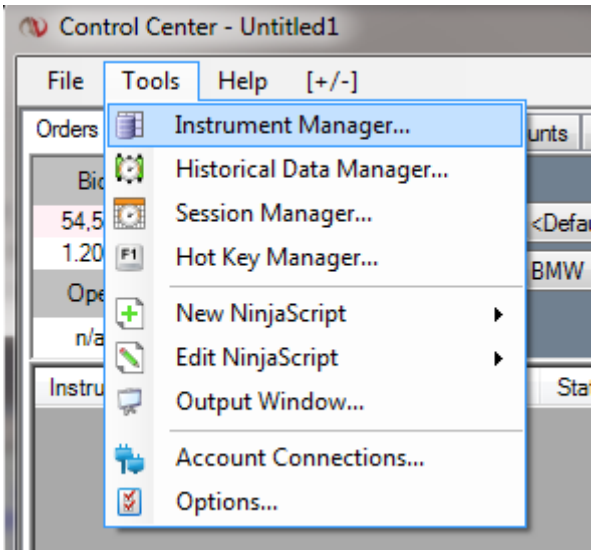
Start TradeRouter. Make sure you have some engines in TradeRouter to receive price data. Otherwise you won't be receiving price data nor in the FrontEnd nor in NinjaTrader.

Now start the FrontEnd application. When up and running, left-click on 'Get List'. The FrontEnd will now connect with TradeRouter and make a request for the engines. Each price tick received from TradeRouter is send to NinjaTrader.

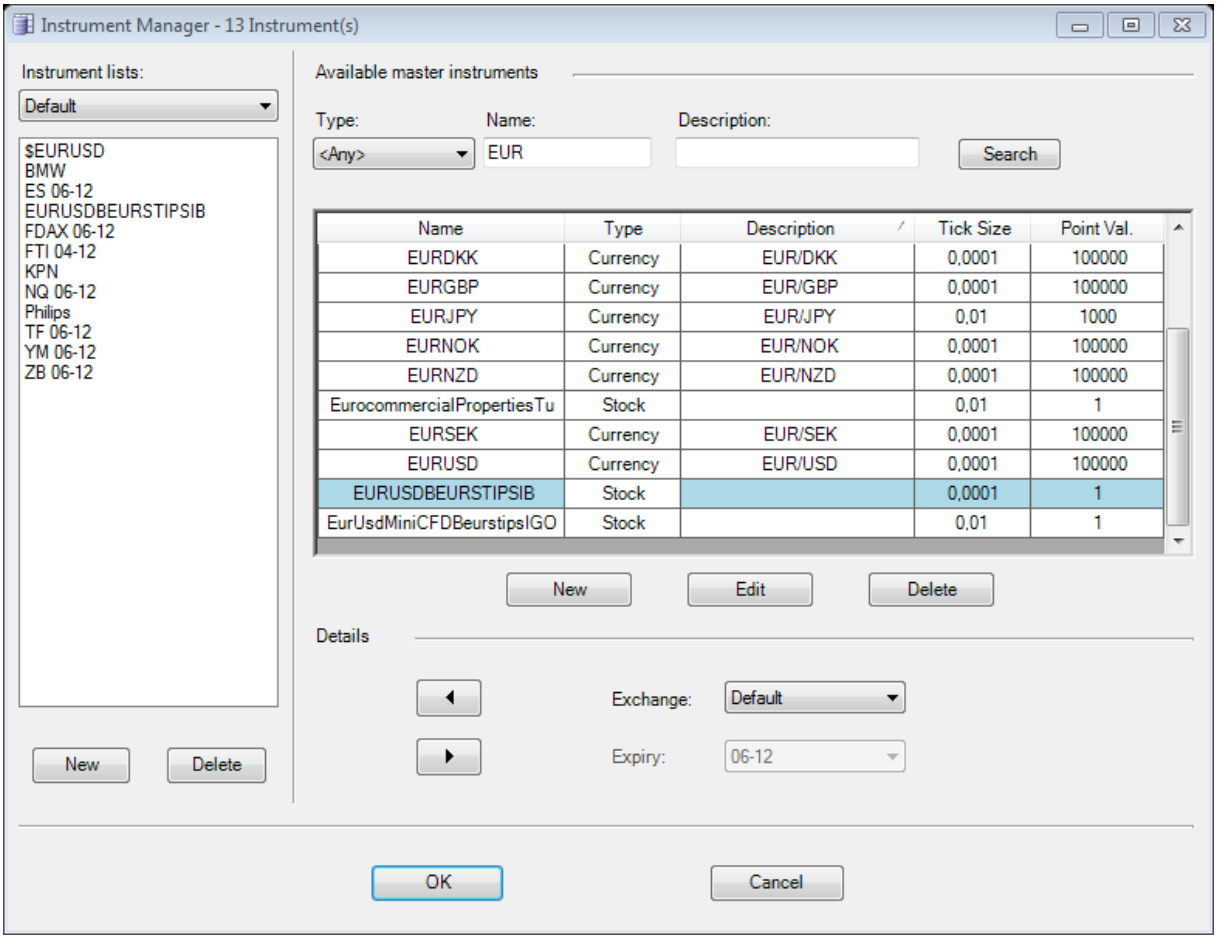


The engine names in TradeRouter will be trimmed before sending them to NinjaTrader. Example: AEX (Beurstips, IB2) in TradeRouter will be shown as AEXBEURSTIPSIB in NinjaTrader. This is needed because of limitations of the ATI of NinjaTrader. The ATI only accepts names with normal characters. It will be capitalized, because when editing an instrument in NinjaTrader you can only use capital letters for its name. 0-9 and punctuations are not allowed. The FrontEnd does this automatically.

To use an engine in NinjaTrader, you will need to add it as an instrument trough the Instrument Manager. Left-click 'Tools->Instrument Manager…'

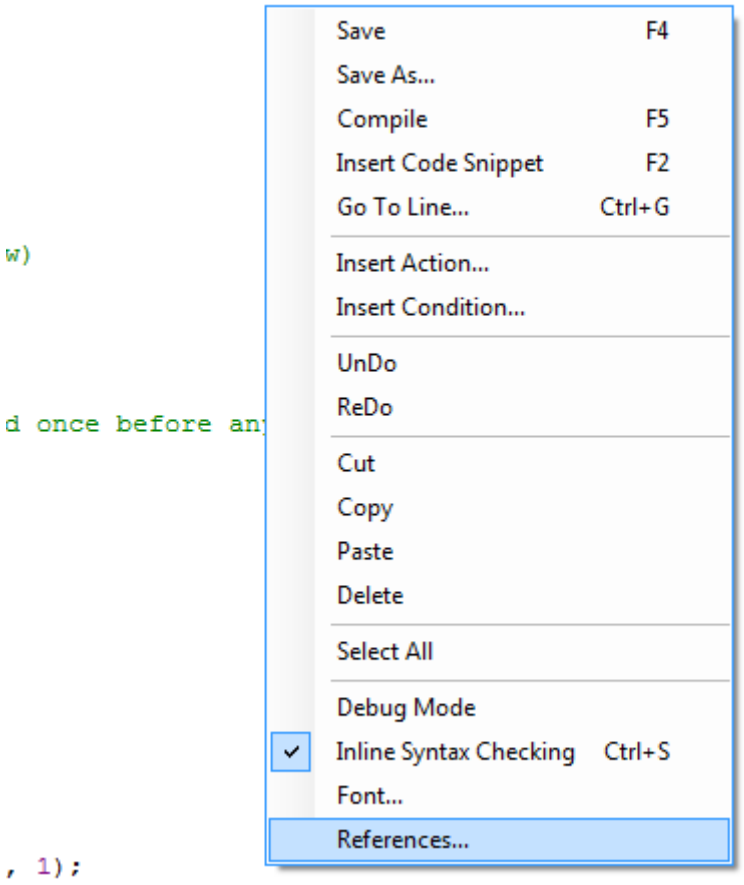


Then search for the name of the engine and select the instrument. Now left-click on the button with the left arrow. Finally click on 'OK'.

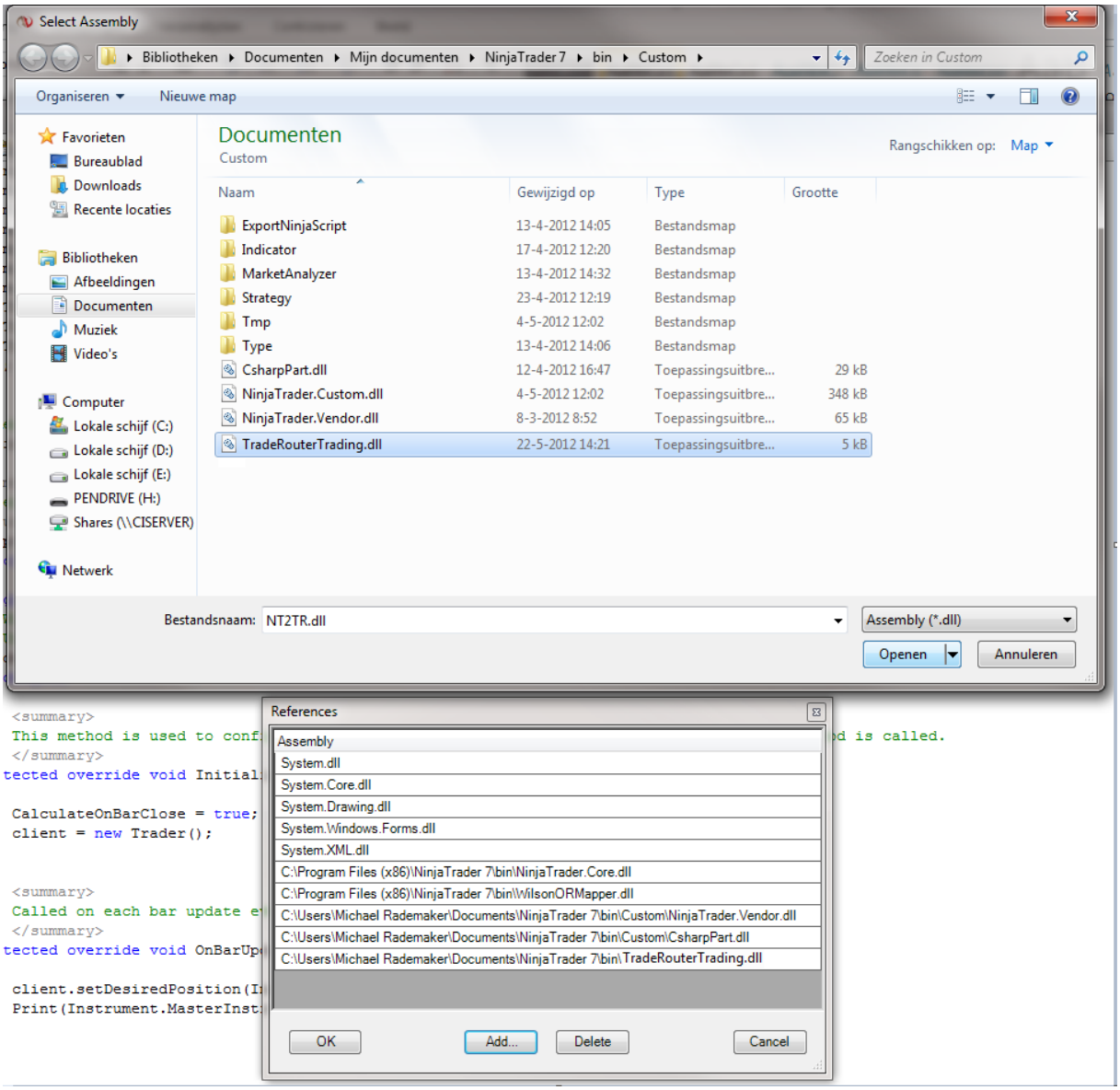


5. NinjaScript

To use the TradeRouterTrading.DLL in a NinjaScript, you need to add a reference. This is done by right-clicking in the script. Then left-click 'References…'



Click on 'Add…' in the next screen. This opens a browse screen. Go to the folder where you've copied the TradeRouterTrading.DLL to.



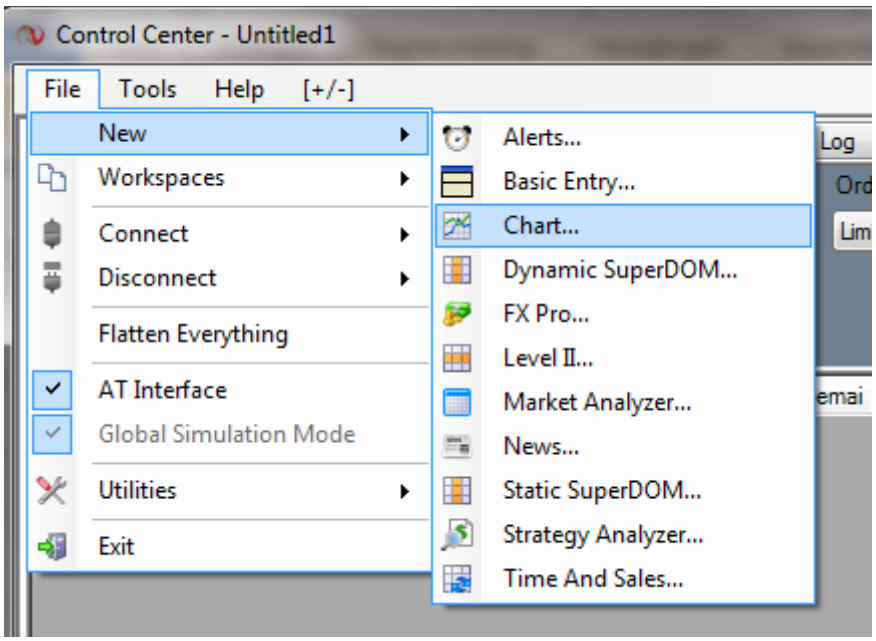
End this step by left-clinking on 'OK'. The DLL is now added as a reference. Type in the Using declaration region NT2TR. This is the namespace of the DLL. You can now start using the DLL in your NinjaScript.

On the next page you'll see a simple example using TradeRouterTrading.DLL. When the script initialises, it makes an instance of the Trader class. The FrontEnd is running locally on this computer, so no parameters are needed.

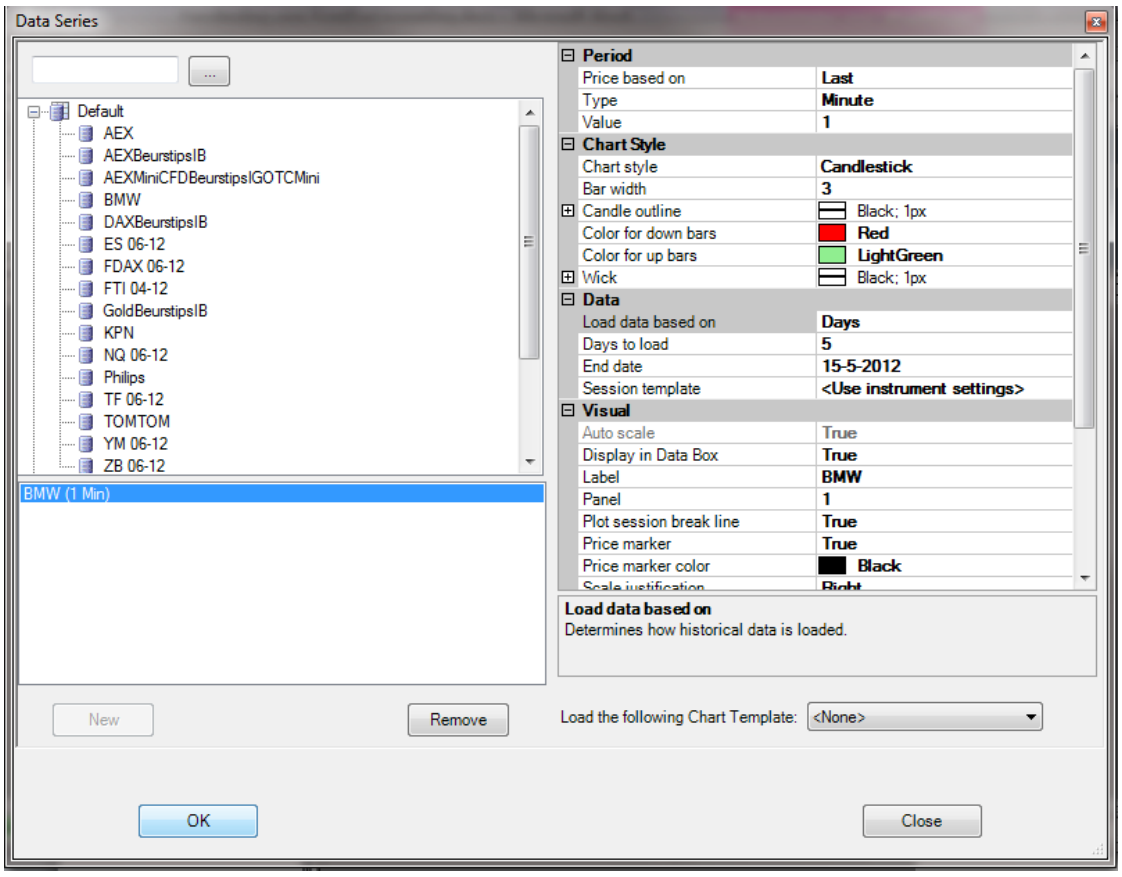
```
#region Using declarations
using System;
using System.ComponentModel;
using System.Diagnostics;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Xml.Serialization;
using NinjaTrader.Cbi;
using NinjaTrader.Data;
using NinjaTrader.Gui.Chart;
using NT2TR;
#endregion

// This namespace holds all strategies and is required. Do not change it.
namespace NinjaTrader.Strategy
{
    /// <summary>
    /// Enter the description of your strategy here
    /// </summary>
    [Description("Enter the description of your strategy here")]
    public class TRTest : Strategy
    {
        #region Variables
        // Wizard generated variables
        // User defined variables (add any user defined variables below)
        Trader client;
        #endregion

        /// <summary>
        /// This method is used to configure the strategy and is called once
before any strategy method is called.
        /// </summary>
        protected override void Initialize()
        {
            CalculateOnBarClose = true;
            client = new Trader();
        }

        /// <summary>
        /// Called on each bar update event (incoming tick)
        /// </summary>
        protected override void OnBarUpdate()
        {
// Go long if the median price of the last 10 open prices is higer than 100
            if (GetMedian(Open, 9) >100)
           {

        client.setDesiredPosition(Instrument.MasterInstrument.Name, 1);
           }
// Go short if the median price of the last 10 open prices is lower than 90
           if (GetMedian(Open, 9) < 90)
           {

        client.setDesiredPosition(Instrument.MasterInstrument.Name, -1);
           }
        }
    }
}
```

6. Adding a NinjaScript to a chart

These steps will show you how to use a strategy (NinjaScript) on an instrument.



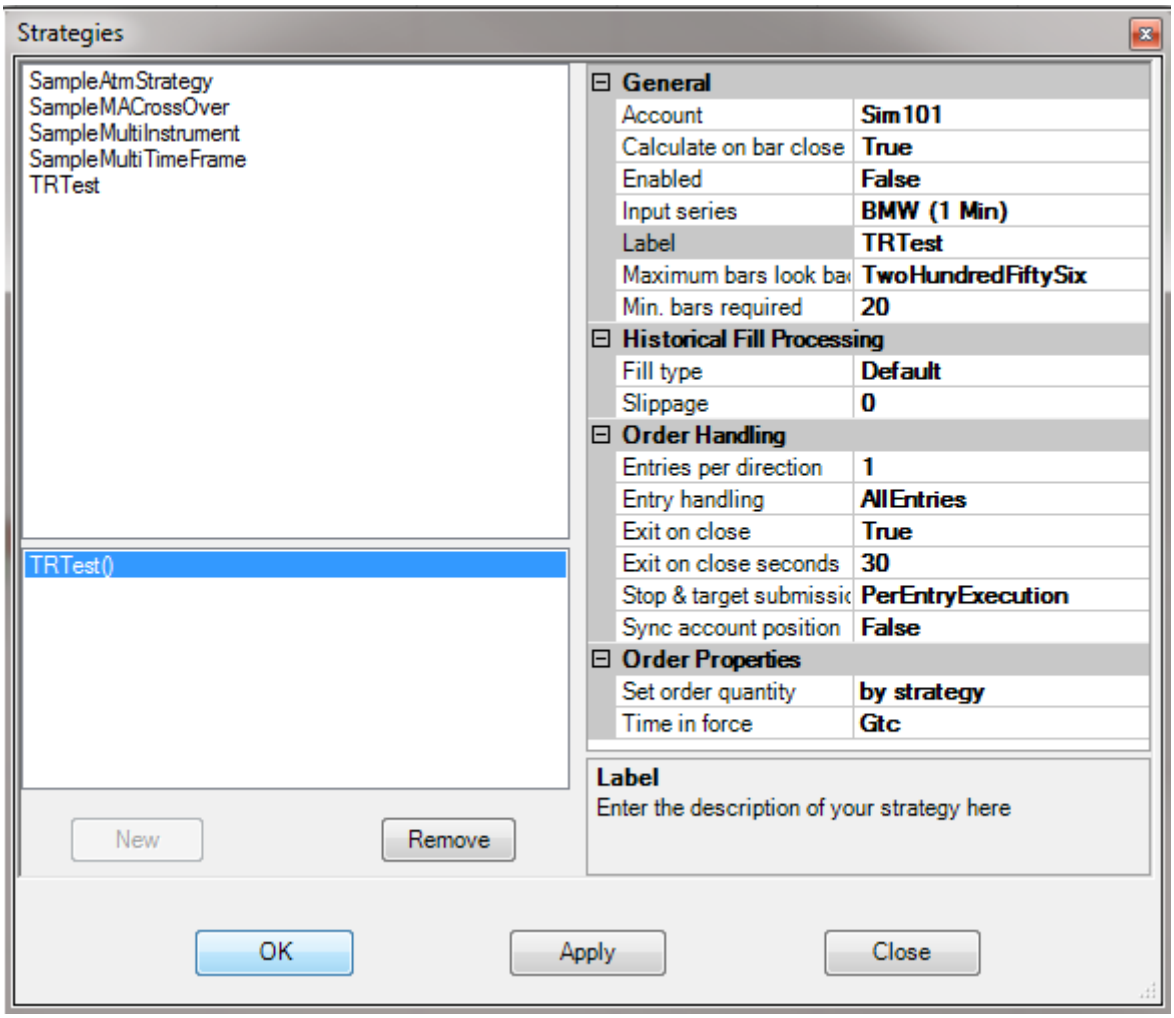
Left-click on 'File->New->Chart...'



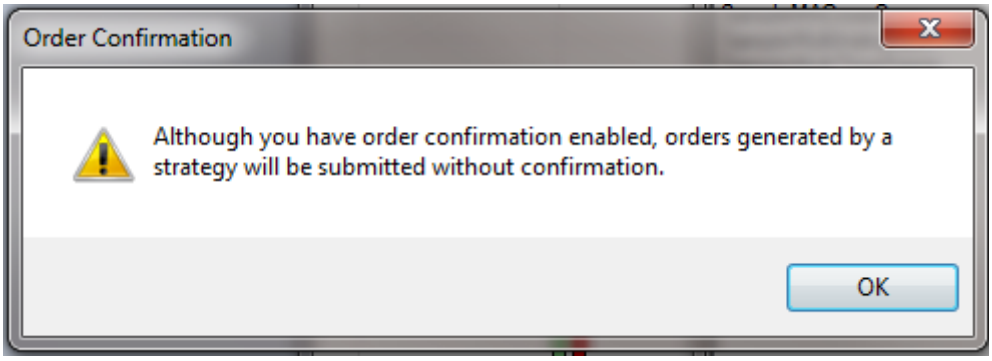
Select the instrument you want to use. This is done by double clicking on the instrument or clicking once and then clicking on 'New'. When selected, you can edit some options on the right. When finished, click 'OK'.

A window will open with a chart in it. You will see price data from the moment the FrontEnd is connected with TradeRouter and NinjaTrader and is sending price data.

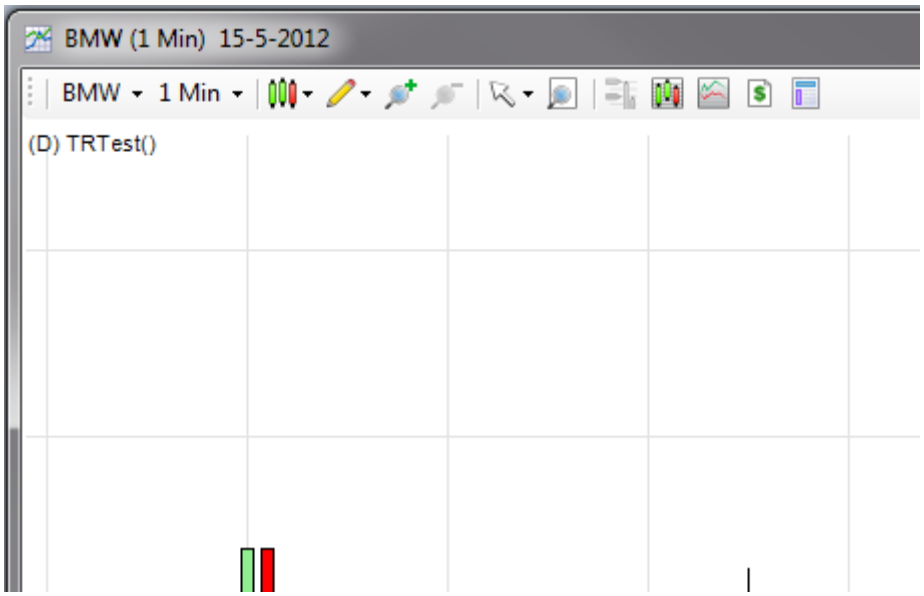
To add a strategy on the chart, left-click on the dollar sign.



Select the strategy you want to use. This is done by double-clicking on the strategy or clicking once and then clicking on 'New'. When selected, you can edit some options on the right. When finishend, click 'OK'.



You will see this window. Click on 'OK'.



You will see the name of the strategy on the top-left corner of the chart window.

Bijlage VIII

Nevenactiviteiten

Nevenactiviteiten

Opdrachten tijdens de afstudeer periode

Tijdens mijn afstudeer periode bij Capital Intervest heb ik niet alleen aan mijn afstudeer opdracht gewerkt, maar ook aan kleinere opdrachten. Dit gebeurde tussendoor, wanneer ik moest wachten op feedback van de bedrijfsbegeleider.

Inhoudsopgave

1. De opdrachten

1.1 ABN-AMRO Turbo feed

ABN-AMRO heeft zelf een financieel product bedacht genaamd Turbo. Dit zijn producten die alleen ABN-AMRO aanbiedt. Ze zijn ook de enige die daar koersinformatie over versturen.

Ik heb verbinding gemaakt met hun webservice. Via die verbinding ben ik gaan onderzoeken welke informatie er beschikbaar is op de webservice. Ik heb in kaart gebracht hoe deze verstuurd wordt en hoe je die het beste kunt opvragen. Dit staat allemaal vastgelegd in een document (bijlage I).

1.2 ING Sprinter feed

De ING heeft ook een eigen financieel product; de sprinter. Ook zij hebben een webservice die daar koersinformatie over verstuurd.

Ook met die webservice heb ik een verbinding gemaakt, onderzoek uitgevoerd en alles vastgelegd in een document (bijlage II).

1.3 Sms dispatcher

Een voormalig stagiair bij Capital Intervest heeft een applicatie gemaakt, waarmee de TradeRouter sms-en kan versturen en ontvangen. Dit gaat via een Engelse provider die een API aanbiedt. Het project was nog niet helemaal af en er zaten nog een paar fouten in.

De applicatie heb ik afgemaakt, naar eigen inzicht verbeterd en de fouten eruit gehaald. Vervolgens heb ik het getest en gedemonstreerd aan mijn bedrijfsbegeleider. Die heeft het uiteindelijk in productie genomen.

1.4 Support

Klanten van Capital Intervest kunnen bellen als ze ergens niet uitkomen of een fout constateren. Als er meerdere bellers zijn en alle werknemers zijn bezet, dan neem ik de telefoon op en noteer hun klacht of vraag. Die klacht of vraag zet ik door naar één van de werknemers die later contact met de klant opneemt om het op te lossen.

2. Bijlage I – ABN-AMRO Turbo feed

2.1 TurboServiceService

De class TurboServiceService heeft geen variabelen, alleen maar methodes om data op te halen.

Methode	Parameters	Return
getQuote	String isin	TurboQuote
getQuotes	String[] isin	TurboQuote[]
getProduct	String isin	TurboProduct
getProducts	String isin[]	TurboProduct[]
getProductsActive		TurboProduct[]
getProductsEnded		TurboProduct[]

De methodes hierboven zijn ook in een Async vorm, dat wilt zeggen dat ze asynchroon kunnen worden uitgevoerd.

Daarbij horen events, om te weten wanneer de data binnenkomt.

Event	Parameters
getQuoteCompleted	Object sender, getQuoteCompletedEventArgs e
getQuotesCompleted	Object sender, getQuotesCompletedEventArgs e
getProductCompleted	Object sender, getProductCompletedEventArgs e
getProductsCompleted	Object sender, getProductsCompletedEventArgs e
getProductsActiveCompleted	Object sender, getProductsActiveCompletedEventArgs e
getProductsEndedCompleted	Object sender, getProductsEndedCompletedEventsArgs e

2.2 TurboQuote

De class TurboQuote kent geen methoden, alleen maar variabelen.

Variabele	Type	Beschrijving
Ask	Float	Vraag prijs
Asksize	Int	
Bid	Float	Bied prijs
Bidsize	Int	
Change	Float	Verandering sinds opening
Current_leverage	Float	Hefboom
Isin	String	Id van de turbo
Timestamp	String	Tijdsstempel

2.3 TurboProduct

De class TurboProduct kent geen methoden, alleen maar variabelen.

Variabele	Type	Beschrijving
Asset__class	String	
Current_knockout_level	Float	
Current_ticker	String	
Exchange	String	De exchange van de Turbo, bijv. Xtra
Fx_rate	Float	
Isin	String	De ID van de turbo
Issue_date	String	Datum van uitgifte
Issuer	String	Instantie die de Turbo uitgeeft
Knockout_payment	Float	
Knockout_timestamp	String	Datum van de knockout
Market	String	De markt waar de turbo zich in bevindt
Next_roll_date	String	
Product_cat	String	Categorie van de turbo bijv. LEV
Product_type	String	Type van het product bijv. MINI
Product_type_description	String	Beschrijving van het type
Ratio	Float	
Reference_currency	String	De valuta voor de turbo
Reference_price	Float	Prijs voor de turbo
Status	Int	Status van de turbo
Strike	Float	
Strike_date	String	
Trading_size	Int	
Type	String	Type van de turbo als in Long of Short
Underlying_isin	String	
Underlying_name	String	Het werkelijke naam bijv. AEX of adidas
Underlying_type	String	Het werkelijke product bijv. Index of Share

Dit bevindt zich allemaal in de .Net omgeving, door de WSDL van ABN-AMRO te importeren als een Web Service. Dit is gebeurd in Visual Studio 2010. De link naar de WSDL is:

http://abn-markets.customers.solvians.com/services/finance-broker/?wsdl

3. Bijlage II – ING Sprinter feed

3.1 ING Sprinters

De ING Sprinters worden aangeboden via een ASPX .Net server. Het is niet mogelijk om het via een Web Reference binnen te halen en er zo gemakkelijk ermee te werken. In het project dat ik heb gemaakt heb ik echter wel een aantal classes gemaakt die wat meer inzicht moeten geven. Inzicht over wat voor data er wordt aangeleverd. Het project bevindt zich in het .Net 4 Framework en niet in het .Net Cliënt Profile. Dit in verband met het gebruiken van de JavaScriptSerializer om de JSON te parsen. De data die ik dan ook heb gebruikt was in JSON formaat. Het project zal het wat duidelijker maken en in de documentatie van de ING staat het ook goed beschreven.

3.2 Sprinter

De class Sprinter kent geen methoden, alleen maar variabelen.

Variabele	Type	Beschrijving
Ask	Float	Vraag prijs
Bid	Float	Bied prijs
Currency	String	Valuta van de sprinter
Leverage	Float	Hefboom
Isin	String	Id van de sprinter
Timestamp	String	Tijdsstempel
Symbol	String	Symbol van de sprinter
productName	String	Naam van het product
productFullName	String	Volledige naam
Product	String	(Sprinter)
ProductType	String	Long of Short
IssueSize	Float	Uitgave grote
IssueDate	String	Uitgave datum
LaunchDate	String	Lanceer datum
Ratio	Float	
Future	String	
FinancingLevel	String	
Fx_Rate	Float	
StopLoss	Float	
StopLossDate	String	
ResidualValue	String	
endDate	String	
Underlying	Underlying	Een object met informatie over underlying elementen
UpdateTime	String	Tijd van laatste update
Status	String	Active of Stopt
Mid	Float	
MidClose	Float	
Verschil	Float	
VerschilP	Float	
ING_ef	String	
Extra1	String	Extra informatie

Extra2	String	Extra informatie

3.3 Underlying

De class Underlying kent geen methoden, alleen maar variabelen.

Variabele	Type	Beschrijving
UnderlyingName	String	Onderliggende naam
UnderlyingIsin	String	Onderliggende isin
UnderlyingRegion	String	Onderliggende regio
UnderlyingType	String	Onderliggende type
UnderlyingCurrency	String	Onderliggende valuta
UnderlyingPrice	Float	Onderliggende prijs

3.4 SmallSprinter

De class SmallSprinter kent geen methoden, alleen maar variabelen.

Variable	Type	Beschrijving
Isin	String	ID van de sprinter
Bid	Float	Bied prijs
Ask	Float	Vraag prijs
SmallUnderlying	SmallUnderlying	Object van SmallUnderlying
UpdateTime	String	Tijd van laatste update

3.5 SmallUnderlying

De class SmallUnderlying kent geen methoden, alleen maar variabelen.

Variable	Type	Beschrijving
Price	Float	Onderliggende Prijs