tgm
Die Schule der Technik

DIPLOMARBEIT

NOCTUA

Wertpapierhandelsalgorithmus mit Marktzustandsadaption

Ausgeführt in Zuge der Reife und Diplomprüfung
Ausbildungszweig Systemtechnik/Medientechnik

unter der Leitung von
Prof. Mag. Hans Brabenetz
Prof. Dr. Helmut Vana
Abteilung für Informationstechnologie

eingereicht am Technologischen Gewerbemuseum Wien
Höhere Technische Lehr- und Versuchsanstalt
Wexstrasse 19-23, A-1200 Wien

von
Peer Nagy 5CHITI
Gabriel Pawlowsky, 5BHITS
Josef Sochovsky, 5BHITS

Wien, im Oktober 2012

Abteilungsvorständin:	Prof. Dipl.-Ing. Grete Kugler
Tag der Reifeprüfung:	xx. xx xxxx
Prüfungsvorsitzender:	Univ.–Prof. Dipl.-Ing. Dr.techn. xxx
Erster Gutachter:	Dipl.-Ing.(FH) Mag. Dr.techn. Gotti Koppi
Zweiter Gutachter:	Prof. Dr.techn. Wenn Vorhanden

Vorwort

Diese Arbeit wurde im Jahr 2012 im Zuge unserer Ausbildung in der Abteilung für Informationstechnologie am Technologischem Gewerbemuseum (TGM), HTBLVA Wien 20, durchgeführt.

Dankesworte

Wien, im Oktober 2012 Name, Name, Name, Name

Abstract

This is the english abstract.

Kurzfassung

Deutsche Kurzfassung kommt hierher

Inhaltsverzeichnis

Abbildungsverzeichnis

Listings

KAPITEL 1

Einführung

1.1 Hintergrund und Motivation

1.1.1 Hintergrund

Seit dem Beginn des 17. Jhdt., wo die ersten Anteilsscheine von Unternehmen ausgegeben wurden hat sich der Finanzmarkt in Riesenschritten vorwärts bewegt und ist heutzutage eine Multi- Milliarden-Dollar Industrie. Aktien sind längst nicht mehr die einzigen Wertpapiere, die auf dem Finanzmarkt gehandelt werden. Instrumente wie Optionsscheine oder Future-Contracts sind dabei noch etabliertere Handelsgüter.

Der Handel mit Wertpapieren ist in den letzten Jahren und Jahrzehnten zunehmend systematisiert und automatisiert worden. Kaum jemand trifft Handelsentscheidungen leichtfertig aus dem Bauch heraus ohne fundierte Analyse. Diese Analyse unterwirft sich aber damit einem programmatischen Schema, das ebenso gut auch automatisch, algorithmisch angewandt werden kann: trifft ein Mensch Entscheidungen nach einem genauen Schema, kann ein Computer dies ebenso und dabei sogar schneller und genauer.
Besonders gut geeignet dafür scheinen die technische Analyse, besonders die Trendbestimmung und das Trendfolgen. Auch wenn es reichlich Kritik an solchen Systemen gibt (besonders, darauf basierend, dass sich Aktienkurse nach keiner bekannten statistischen Verteilung bewegen), wenden sehr viele Marktteilnehmer solche Systeme an. Das führt zumindest teilweise aber zu einer selbsterfüllenden Prophezeiung, da sich die Kurse am Verhalten der Majorität der Marktteilnehmer orientieren.

Ein weiterer Vorteil des Algorithmischen Trading ist die Geschwindigkeit, sowie Genauigkeit mit der Computer arbeiten können, an die Menschen nicht heranreichen. Durch systematische und statistische Entscheidungen können menschliche Emotionen aus dem Spiel gelassen und dadurch auch das Risiko besser abgeschätzt werden.

Die Informationsflut, die schon für normale Aktien existiert, sind längst nicht mehr manuell zu bewältigen. Räume voller Server rechnen ununterbrochen mit den Kursdaten und versuchen jeden möglichen Vorteil auszunutzen, um den Gewinn zu optimieren. Die verwendeten Modelle, Ansätze und Algorithmen werden in der Regel geheim gehalten, da viel Geld von Entscheidungen der Software abhängt. Wären verwendete Algorithmen publik, würden diese nicht mehr lange profitabel funktionieren. Ergo bleiben die Vorgangsweisen, die besser funktionieren und großen Unternehmen reale Gewinne einbringen geheim. Im Internet gibt es viele Quellen, die verschiedene Ansätze erklären, wie mit vorhandenen Daten Handelsentscheidungen berechnet werden können, die in Tradingsoftware implementiert werden könnten. Diese Standardalgorithmen sind au mass vorhanden, ein objektiver Vergleich ist aber entweder sehr umständlich oder gar nicht möglich. Ein besonderer Fokus dieses Projektes soll daher darauf liegen entwickelte Algorithmen auf ihre Performance zu prüfen. Zu diesem Zweck wird eine BTS programmiert, welche die Signale eines Algorithmus für einen historischen Datenbestand generiert und die resultierenden virtuellen Trades analysiert.

Momentan haben sowohl Börseninteressierte als auch professionelle Trader ein Problem damit, die Performance unterschiedlicher Algorithmen adäquat zu klassifizieren und damit auch zu vergleichen. Das riesige Spektrum an algorithmischer Tradingsoftware, das derzeit angeboten wird, ist zudem auch noch sehr undurchsichtig, da es sich bei fast allen arbeitenden Algorithmen um closed Source Produkte handelt. Der normale Benutzer kann beim Kauf einer solchen Software also nur die Kompetenz des anbietenden Unternehmens einschätzen und hat keine schriftliche und überprüfbare Grundlage, die die Qualität des Algorithmus aufzeigt, in dessen Hände er sein Geld legt.

Am einfachsten kann hierbei durch die Entwicklung eines eigenen oder durch den Kauf des Source Codes eines anderen Algorithmus Abhilfe geschafft werden. Ohne die Noctua-BTS wäre es jedoch trotzdem nicht möglich, solche Algorithmen auf ihre Performance oder auch auf die Arbeitsweise während unterschiedlichen Marktzuständen zu testen.
Zusätzlich fixiert sich die absolute Mehrheit der derzeit vorherrschenden Algorithmen zu stark auf die reine Kursentwicklung und beachtet nicht den aktuellen Marktzustand, obwohl dieser einer der wichtigsten Grundsteine zur Vorhersage der Kurse ist.
Natürlich ist es außerdem zurzeit nicht möglich sein Kapital so fixverzinst an-

zulegen, dass man einen Jahreszinssatz von 10% erhält. Am nächsten kann man dem nur durch die sehr risikoarme Verwaltung seines Kapitals durch einen ausgeklügelten Algorithmus kommen, der genau dies verspricht.

1.1.2 Motivation

Für ein automatisiertes Handeln mit Wertpapieren an der Börse soll ein Algorithmus entwickelt werden, der die optimalen Handelsentscheidungen berechnet. Damit man den fertigen Algorithmus, sowie dessen Vorgängerversionen, als auch andere Standartalgorithmen (wie zum Beispiel:) miteinander vergleichen kann, ist es erforderlich eine Software zu implementieren die automatisch mit historischen Daten, ergo bereits vergangene Bewegungen an der Börse, rechnet und eine vernünftige Testumgebung erzeugt.

Die Bestandteile des Projektes werden in 3 Bereiche gegliedert:

- Der Algorithmus soll Trends möglichst früh identifizieren und diesen solange folgen bis sie durch Support- und Resistance-Level ihre Nachhaltigkeit verlieren. Diese Identifikation soll, mithilfe historischer Daten funktionieren. Damit mit man die Qualität der Algorithmen in verschiedenen Marktzuständen die jeweilige Qualität der Berechnungen feststellen kann, ist es notwendig den Markttrend zu gewissen Zeiten bereits zu kennen. Das bedeutet, dass die vorhandenen Daten analysiert werden müssen, um die verschiedenen Marktumschwünge und deren Auswirkungen auf den Algorithmus zu erkennen und auslesen zu können.

- Um die verschiedenen Kursmuster bei diversen Marktstimmungen in den Algorithmus zu integrieren, unterscheidet dieser zwischen Marktzuständen, die durch gesamtwirtschaftliche Zusammenhänge (Währungswechselkurse, Commodity-Preise, Leitindizes, u.Ä.) bestimmt werden können. Hierbei muss besonders vorsichtig mit den Änderungen an dem Algorithmus vorgegangen werden, sonst kann es leicht passieren, dass eine kleine Änderung der Grundberechnung die Entscheidung bei einer völlig unerwarteten Trendänderung beeinträchtigt. Deswegen ist es wichtig jede Änderung zu dokumentieren und zu versionieren. Dies soll einerseits das Endergebnis der Entwicklung verbessern und qualitativ hochwertiger halten und andererseits eine signifikante Geschwindigkeitssteigerung in der Algorithmuskodierung erzeugen.

- Damit die Performance auch über mehrere Versionen des Algorithmus verglichen werden kann, soll eine BTS entwickelt werden, diese wird mithilfe der historischen Daten funktionieren. Dieses Programm simuliert einen rapiden Ablauf von möglicherweise mehreren Jahren Börsengeschichte, die

den Rechenablauf mit verschiedenen Marktzuständen und Marktzustandsänderungen konfrontiert. Die Ergebnisse dieser Simulation können dann ausgelesen werden, um sie mit den vorherigen Ergebnissen auf eine etwaige Steigerung oder auch einen Abfall der Performance zu analysieren. Damit man einen reibungslosen Austausch der verschiedenen selbst entwickelten Rechenmodulen, aber auch der vorhandenen Standardalgorithmen anbieten kann, muss dafür gesorgt werden, dass das simple und effiziente Austauschen des momentan verwendeten Algorithmus ermöglicht wird.

Noctua soll es vereinfachen verschiedene Ideen, die teils auf dubiosen Webseiten angepriesen werden, mit einander vergleichbar zu machen und den möglichen Gewinn und das Risiko zu klassifizieren. Dazu muss nur noch der Algorithmus nach vorgegebener Art und Weise implementiert und von der BTS getestet werden.
Durch die Entwicklung des Noctua-Algorithmus werden außerdem eine Reihe von Ideen ausgetestet, wodurch ebenfalls die Performance der angewendeten Indikatoren und damit verbundenen Tradingstrategien verifiziert werden kann. Zusätzlich entsteht durch das inkrementelle Voranschreiten der Algorithmus-Version objektiv messbares Wissen über finanzwirtschaftliche Zusammenhänge im Bereich des algorithmischen Trading.

1.2 Objectives of this Thesis

1.3 Methodology for the Developement

1.4 Thesis Outline

KAPITEL 2

Machbarkeitsstudie

2.1 Voruntersuchung

2.1.1 IST-Erhebung

Es wird bereits ein Großteil des Kapitals in Wertpapieren algorithmisch verwaltet. Daher existiert eine Unmenge an Wissen über den Aufbau von Börsenalgorithmen und die Anwendung von Indikatoren zur Einschätzung von zukünftigen Kurswerten. Die meisten dieser Algorithmen basieren auf technischer Analyse und den simplen Indikatoren die diese mit sich bringt. Bekannte Vertreter davon sind zum Beispiel der Moving Average Convergence Divergence (MACD) und der Commodity Channel Index (CCI). Die meisten Algorithmen benutzen außerdem eine Zusammensetzung aus verschiedenen Moving Averages (MAs), um die zu Grunde liegenden Handelsentscheidungen zu treffen. Aufgrund dieses umfangreichen Wissens ist es möglich weitere Möglichkeiten zu erforschen und noch besser und sinnvoller handelnde maschinelle Helfer zu kreieren. Weniger umfangreich ist allerdings das Wissen über Marktzustände. Es gibt eine Menge Aufzeichnungen über die simplen Marktphasen(Aufwärts-, Abwärts-, Seitwärtstrend), doch Methoden zur Kategorisierung des Marktes in neue Klassen sind eher wenig vorhanden bzw. schlecht oder nur unternehmensintern zugänglich.

Ein Problem der aktuellen Situation ist allerdings, das schlechte bzw. umständliche Testing dieser Algorithmen, da wenig Software existiert, die verifizieren kann ob ein Algorithmus in bestimmten Marktphasen bestimmte Leistungen erbringt. Außerdem ist es momentan ziemlich kompliziert sich einfach die gesamte Performance eines solchen Handelsalgorithmus anzusehen. Es gibt hierfür aber sowohl Gratisquellen, als auch kommerzielle Produkte, von denen man historische Daten zum Backtesting dar Algorithmen beziehen kann. Das Projektteam von Noctua besitzt bereits ca. 3.9 Gb an historischen Börsenkursen von e-Signal [1] und nahezu unbegrenzten Zugriff auf weitere Daten vom selben Anbieter. Dadurch ist es ihm möglich Algorithmen über ein weites Spektrum von Marktphasen und -zuständen hinweg zu testen und die Performance verschiedener Algorithmen in unterschiedlichen Situationen akkurat festzustellen. Dies ist besonders wichtig, da Algorithmen die in der nahen Vergangenheit guten Entscheidungen trafen, meist weiterhin sehr erfolgreich handeln und den Anleger möglicherweise mit einem Kapitalzuwachs belohnen.

2.1.2 IST-Zustand

Aufgrund der riesigen Industrie die diesem Projekt zu Grunde liegt gibt es natürlich bereits eine Vielzahl an Konkurrenzprodukten auf dem Markt. Einige davon spezialisieren sich auf das Backtesting bzw. die Bereitstellung einer Plattform zur Entwicklung von Algorithmen. Andere sind eher proprietärer Natur und

[1] http://www.esignal.com/default.aspx?tc=

Abbildung 2.1: MetaTrader 5 Oberfläche

versuchen lediglich durch Korruption der Konkurrenz, selbst den wirtschaftlich rentabelsten Algorithmus zu betreiben. Doch alle haben ihre Vor- und Nachteile. Daher finden sich auch immer wieder neue Nieschen für Neueinsteiger am Markt, die durch ausgeklügelte Algorithmen viel erreichen können. Im folgenden werden nun die bekanntesten dieser Konkurrenzprodukte beschrieben.

2.1.2.1 MetaTrader 5

Hierbei[2] handelt es sich um ein ziemlich umfangreiches, ein wenig älteres Produkt, dass sowohl Aktien- als auch Forex-Daten anbietet. MetaTrader 5 ist die neue verbesserte Version von Metatrader 4 und bietet neben den alten Funktionen, Neuerungen wie die Einbindung von News. Außerdem bietet der MetaTrader ein weites Spektrum an Indikatoren, die einfach in den laufenden Betrieb eingebunden werden können. Die Oberfläche des MetaTraders sieht in etwa aus, wie in Abbildung 2.1 dargestellt.
Doch das größte Feature des MetaTraders ist die eingebaute IDE (siehe Abbildung 2.2), die es mittels einer eigenen MetaTrader-spezifischen Sprache, der MetaQuotes Language 5 (MQL5), ermöglicht eigene Algorithmen zu programmieren und diese dann auch direkt in den laufenden Betrieb zu übernehmen. Der MetaTrader bietet sogar einen jährlichen Wettbewerb an, bei dem er jedes Jahr einen der besten Algorithmen zum Sieger kürt. Die neue Sprache MQL5 ist sogar ebenfalls noch weiter gegenüber der MQL4 verbessert. Dies führt uns allerdings auch

[2] http://www.metatrader5.com/

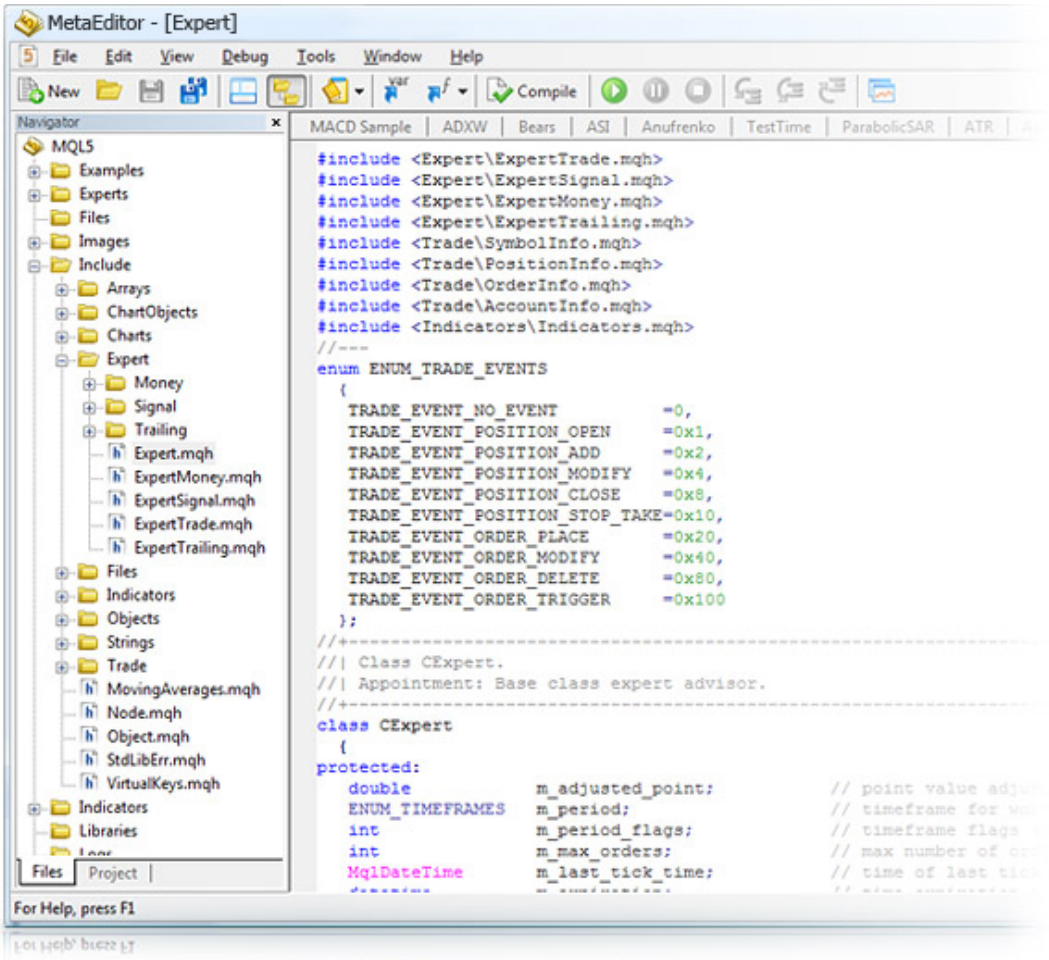

Abbildung 2.2: MetaTrader 5 IDE-Oberfläche

schon zum Problem beim MetaTrader 5, da nur ein geringer Teil der Software Entwickler gewillt ist, wirklich eine neue Programmiersprache zu lernen, nur einen Algorithmus entwickeln und testen zu können, der dann selbst wiederum auch an den MetaTrader gebunden ist nicht in den normalen operativen Betrieb portiert werden kann. Außerdem ist der MetaTrader eines der sehr wenigen Produkte am Markt, die es Nutzern wirklich ermöglicht einen Algorithmus zu entwickeln und zu testen ohne die gesamte Struktur rundherum zuerst aufzubauen. Daher ist es nötig diesem Mangel Abhilfe zu schaffen und eine BTS zu entwickeln, die genau dies ermöglicht und ohne den Nutzer dabei an das Unternehmen in dem er seinen Algorithmus testet zu binden.

2.1.2.2 Algorithmen

Es existieren wie bereits des öfteren erwähnt Unmengen an proprietären Algorithmen, die Zahl der Open Source Algorithmen hält sich allerdings in Grenzen. Daher ist es relativ schwierig die Konkurrenzalgorithmen fundiert zu analysieren. Ein guter Ansatz, um herauszufinden wie diese proprietären Algorithmen wirklich handeln, ist sich die bekanntesten Einzelmethoden anzusehen, da höchstwahrscheinlich ein Großteil der proprietären Algorithmen aus diesen Einzelmethoden unterschiedlichster Art aufgebaut sind. Wahrscheinlich beachten sehr viele Konkurrenzprodukte die unterschiedlichsten Kombinationen aus MAs und schmücken diese mit einer Kombination aus den verschiedensten Börsenindikatoren der technischen Analyse aus. Doch die Wenigsten betrachten den genauen Marktzustand in dem sie sich momentan zur Zeit des Handelns befinden. Wie bereits erwähnt sind nur die simpelsten Marktzustände öffentlich bekannt und zugänglich und diese sind schon sehr benützt. Sie werden von einer so großen Anzahl an Softwareprodukten benutzt, dass man keinen oder nur mehr wenig Vorteil mehr daraus zieht, sie zu beachten. Deswegen liegt die einzige Möglichkeit in der Verbesserung von Software in diesem Punkt in der Entwicklung von eigenen Klassen von Marktzuständen. Wählt man hier richtig aus und investiert ausreichend Zeit in die Forschung, so kann man hiermit leicht an der Konkurrenz vorbeiziehen und performantere Algorithmen kreieren.

2.2 Produktfunktionen

/LF10/
Vergangene Marktzustände bestimmen

Beschreibung	Es sollen historische Marktzustände (innerhalb der letzten Jahre), auf transparenten Aktienmärkten, für die ein ausreichender Datenbestand vorhanden ist, automatisch bestimmt werden. Sollten sich verschiedene große Märkte entgegen Erwartung entsprechend unterschiedlich verhalten, dass diese keiner einheitlichen Analyse unterzogen werden können, soll primär der US-amerikanische Aktienmarkt untersucht werden. Hierbei handelt es sich um eine Gruppierung von Zeitabschnitten nach gemeinsamen Kriterien.
Aufwand	Hoch
Nutzen	Hoch
Ziel	Ermittelung von Klassen für Marktzustände.
Vorbedingungen	-
Nachbedingungen	-

/LF20/
Aktuellen Marktzustand bestimmen

Beschreibung	Dabei soll darauf geachtet werden, dass für eine frühe Erkennung möglicherweise nur ein Teil der Daten vorhanden ist, die für die historische Analyse herangezogen werden.
Aufwand	Mittel
Nutzen	Hoch
Ziel	Zuordnung des aktuellen Marktzustandes zu einem bereits Bekannten.
Vorbedingungen	/LF10/ Vergangene Marktzustände bestimmen
Nachbedingungen	-

/LF110/
Trends erkennen

Beschreibung	Durch MAs soll es möglich sein Trends in Aktienkursen zu identifizieren. Dazu kommen verschiedene Crossover-Verfahren (double- / triple-crossover) oder Indikatoren, wie der MACD (Moving Average Convergence Divergence) in Frage. Es soll eine statistisch möglichst profitable Variante hierfür gefunden werden, die aufscheinende nachhaltige Trends möglichst günstig erkennt.
Aufwand	Hoch
Nutzen	Hoch
Ziel	Frühzeitige möglichst profitable Erkennung von Trends.
Vorbedingungen	-
Nachbedingungen	-

/LF120/
MA-Dauer bestimmen

Beschreibung	Je nachdem, wie lange ein Trend andauert, bedingt eine Trenderkennung andere MA(-Paare) mit unterschiedlichen Laufzeiten. Durch Backtesting sollen viele verschiedene Varianten automatisch getestet werden können, um den statistisch besten Parametersatz zu ermitteln.
Aufwand	Niedrig
Nutzen	Mittel
Ziel	Erarbeitung eines optimalen Parametersatzes für ein MA-Paar.
Vorbedingungen	-
Nachbedingungen	-

/LF130/
An Marktzustand anpassen

Beschreibung	Der Algorithmus soll sich durch Parameterveränderung an den erkannten Marktzustand zur Optimierung der Performance anpassen. Dies kann beispielsweise durch verändern der MA-Paare oder durch Anpassung der Market Exposure und damit des Risikos erfolgen. Dazu *können* die Implikationen durch Nachforschung bekannt sein, woraufhin ein Modell angewandt wird, müssen aber nicht, da auch induktiv aus den Implikationen gelernt werden kann, wonach automatisch ein Modell entsteht. (*Maschinelles Lernen*) Dabei werden für die unterschiedlichen Markzustände verschiedene Parametersätze durchprobiert.
Aufwand	Hoch
Nutzen	Hoch
Ziel	Anpassung der Hauptfunktionen des Algorithmus an den aktuellen Marktzustand.
Vorbedingungen	/LF020/ Aktuellen Marktzustand bestimmen
Nachbedingungen	-

/LF140/
Signale generieren

Beschreibung	Signalgeben bei potentiellen Einstiegspunkten (long signal) und Ausstiegspunkten (short signal).
Aufwand	Niedrig
Nutzen	Hoch
Ziel	Rückgabe von Handelssignalen.
Vorbedingungen	/LF110/ Trends erkennen, /LF130/ An Marktzustand anpassen, /LF120/ MA-Dauer bestimmen
Nachbedingungen	/LF160/ Signale filtern

/LF150/
Trend-Nachhaltigkeit bestimmen

Beschreibung	Durch geeignete Support- und Resistance-Indikatoren soll die Nachhaltigkeit eines Trends bestimmt werden (beispielsweise Pivot Points, RSI, CCI oder MAs), um den Ausstiegspunkt zu optimieren.
Aufwand	Mittel
Nutzen	Hoch
Ziel	Festellen der Nachhaltigkeit erkannter Trends.
Vorbedingungen	/LF140 Signale generieren
Nachbedingungen	-

/LF160/
Signale filtern

Beschreibung	Zur Verminderung von unprofitablen, zu kurzen Trades sollen insbesondere Kaufsignale gefiltert werden. Die Trenderkennung könnte des Öfteren zu kurz anhaltende Trends erkennen, indem beispielsweise ein MA-Crossover nur für kurze Zeit besteht. Durch das Einführen eines Schwellenwertes (threshold), der überschritten werden muss, oder eine bestimmte Zeitspanne, die ein Signal überdauern muss können zu kurze Trades vermindert werden, wenn sich im Backtesting dadurch ein Vorteil herausgestellt hat.
Aufwand	Mittel
Nutzen	Hoch
Ziel	Filterung der unbrauchbaren Signale.
Vorbedingungen	/LF140/ Signale generieren, /LF150/ Trend-Nachhaltigkeit bestimmen
Nachbedingungen	-

/LF210/
Performance berechnen

Beschreibung	Die relative Performance eines Algorithmus soll in Prozent der Kapitalveränderung berechnet werden.
Aufwand	Mittel
Nutzen	Hoch
Ziel	Berechnung der relativen Performance eines Algorithmus
Vorbedingungen	-
Nachbedingungen	-

/LF220/
Gewinn/Risiko-Verhältnis berechnen

Beschreibung	Bestimmung des Risikos des Algorithmus (beispielsweise anhand der Volatilität) in Verbindung mit der Performance (e.g. sharpe ratio).
Aufwand	Niedrig
Nutzen	Mittel
Ziel	Bestimmung des Gewinn/Risiko-Verhältnisses eines Algorithmus
Vorbedingungen	/LF210/ Performance berechnen
Nachbedingungen	-

2.3 Technische Machbarkeit

2.3.1 Variantenbildung

2.3.1.1 Programmiersprachen

Die BTS kann man in jeder erdenklichen Programmiersprache schreiben, allerdings ist es wichtig daran zu denken, dass das Programm einerseits effizient arbeiten soll und deswegen hardwarenahe rechnet, und andererseits hat das Projektteam mit manchen Programmiersprachen keinerlei Erfahrung.
Die allgemeine Funktionalität muss das lesen und schreiben von Dateien sein, aber auch das algorithmische Rechnen soll effizient funktionieren. Für das Team kommen daher 3 Möglichkeiten in Frage: Eine Lösung in reinem C++, welches sehr hardwarenahe arbeitet, eine Mischung aus F# und C#, mit der eine parallelisierte Berechnung möglich wäre, und eine Java-Lösung, bei der das Team die größte Erfahrung mitbringt.
Bei der Kombination agiert C# als Handlungs- und Steuerkern und F# als funktionale Programmiersprache, als Rechenkern und Mastermindder Applikation, welches die Entscheidungen trifft. Hierbei wird einerseits eine enorm hohe Arbeitsgeschwindigkeit ermöglicht, da die beiden Sprachen relativ hardwarenah agieren und andererseits besteht der nicht zu unterschätzende Vorteil bzw. die Möglichkeit, den Rechenkern auf ein externes System outzusourcen, welches zum Beispiel enorme Rechenkapazitäten aufweisen könnte und somit viel komplexere und effizientere Algorithmen in annehmbarer Zeit durchrechnen und abhängig davon mehr gewinnbringende Entscheidungen treffen könnte. Dabei sollte es auch bei späteren Erweiterungen des Programms zu keinem signifikanten Geschwindigkeitsabfall kommen.

		Gewichtung	**C++ R*G**		**Java R*G**		**C/F R*G**	
Einfachheit	Aufwand Coding	10%	3	30	1	10	2	20
	Bedienung/ Wartung	6%	3	9	2	6	1	3
	Update	3%	3	9	2	6	1	3
	Integration	5%	3	15	2	10	1	5
	Kenntnisse	6%	3	18	1	6	2	12
	Gesamt	30%	3	90	2	44	1	46
Leistung	Übertragungszeit	6%	1	6	3	18	2	12
	Absturzsicherheit	5%	1	5	2	10	3	15
	Ressourcenverbrauch	3%	1	3	3	9	2	6
	Datenumfang	1%	1	1	3	3	2	2
	Gesamt	15%	1	15	3	50	2	35
Kosten	Lizenzen	10%	1	10	1	10	1	10
	Support	5%	3	15	1	5	2	10
	Betriebskosten	5%	1	5	1	5	1	5
	Dokumentation	5%	1	5	2	20	3	15
	Gesamt	15%	1	30	1	40	2	40
Dokumentation	Verfügbarkeit	10%	3	30	2	20	1	10
	Vollständigkeit	10%	3	30	2	20	1	10
	Qualität	10%	2	20	1	10	1	10
	Gesamt	30%	3	80	2	50	1	30

Kapitel	Gewichtung	**C++**		**Java**		**C#/F#**	
Einfachheit	30%	3	90	2	44	1	46
Leistung	15%	1	15	3	50	2	35
Kosten	15%	2	30	2	40	1	40
Dokumentation	30%	3	80	2	50	1	30

Gesamtbewertung			
Endreihung	3	2	1

Aus der Nutzwertanalyse kann man entnehmen, dass die C#/F# Kombination als die favorisierte Möglichkeit ausgeht, weitere Vorteile die sich aus der Wahl

dieser Mischung ergeben sind: gute Kenntnisse der Programmiersprachen, tolle Community und die Einfachheit, sowie die Erweiterbarkeit. Bei dieser Lösung wird die Steuereinheit vom C# Teil des Programms übernommen und die Rechenaufgaben werden von dem F# Teil bearbeitet. Außerdem ist das .net-Framework sehr beliebt, deswegen kann man damit rechnen das bei einem Problem genügend Helfer gefunden werden können.

2.3.1.2 BTS

Bei der BTS handelt es sich wie bereits erwaehnt, um eine Software, die Algorithmen auf ihre Performance und weitere wichtige Kriterien testet. Um diese Aufgabe zu lösen, müssen zwei wichtige Entscheidungen zur Art der Realisierung getroffen werden:

- Ist es sinnvoller diese Software über eine Graphical User Interface (GUI) steuern zu können, oder reicht es wenn sie in der Konsole arbeitet?
- Auf welche Art und weise soll der Algorithmus der getestet werden soll, zur Laufzeit in die Software eingebunden werden.?

Gehen wir nun also zuerst der Frage nach dem User-Interface nach.
Bei einem großen Anteil der Zielgruppe dieser BTS handelt es sich um Chartisten oder andere sehr visuelle Personen. Diese würden vermutlich eine graphische Oberfläche bevorzugen, da sie dabei z.B. auch ihren Algorithmus so erweitern könnten, dass dieser die Charts die ihm zu Grunde liegen in einem neuen Fenster darstellt und man dadurch einen noch besseren Überblick über das Geschehen des Algorithmus erlangen könnte. Andererseits wird es sich aber auch bei einem großen Teil der Zielgruppe, um Technische Programmierer und Mathematiker handeln, die wieder nicht so viel Wert auf eine graphische Oberfläche legen, da sie eher in Zahlen, als in Bildern denken. Allerdings wäre mit Sicherheit fast niemand dieser Gruppe strikt gegen eine Graphische Oberfläche. Es ist hierbei nur abzuwiegen, ob der erhöhte Aufwand den die Codierung einer solchen GUI mit sich bringen würde, sinnvoll ist und wirklich eine höhere Anzahl an Kunden interessieren würde. Mit einer GUI wäre es allerdings auch erheblich einfacher, einen Algorithmus einzubinden.

Es erscheint dem Projektteam aufgrund der Programmiersprachenwahl am sinnvollsten, dass die User den Algorithmus in der Sprache F# schreiben und diesen als Dynamic Link Library (DLL) konvertieren müssen, um ihn in die Software zu integrieren. Grundsätzlich ist es dadurch in jedem Fall notwendig, dass mit der BTS als Download zusätzlich ein Interface(zur Beschreibung von Methoden-Namen die benutzt werden, usw.) und eine Beispiel-DLL bereitgestellt werden, damit die Nutzer der Software einen schnellen und einfachen Überblick bekommen, wie sie ihre DLLs aufbereiten müssen, um diese problemlos die BTS integrieren zu können. Durch die Frage, wie die Integration eine solchen DLL nun

wirklich am einfachsten für unsere zukünftigen Benutzer wäre, ergeben sich zwei Möglichkeiten zur Aufbereitung und Einbindung eines Algorithmus. Man könnte den Algorithmus einfach zur Laufzeit in die Software laden, indem z.B. die Software ein Dateisystem-Browsing anbietet, indem man die Algorithmus-DLL einfach auswählt und diese wird automatisch in die Software integriert und so der Algorithmus getestet. Würde die Software als Konsolenapplikation laufen, wäre dies schon etwas komplizierter für unerfahrene Nutzer, da man ein bestimmtes Kommando und den direkten Pfad zur Algorithmus-DLL selbst eintippen müsste. Eine ganz andere Möglichkeit wäre es den Algorithmus als eigenständige Applikation bereitzustellen, auf die man über Sockets eine Verbindung aufbauen und die Rechenoperationen an sie auslagern kann. Auf diese Art und Weise wäre es ebenfalls möglich den Algorithmus zu testen, allerdings müsste das Projektteam eine viel Umfangreichere Beispiel-DLL bereitstellen, in der die Eigenständigkeit der Software, sowie die Erreichbarkeit über Sockets direkt nach ihrem Start bereits vordefiniert sein müssten. Dies würde zu einem etwas erhöhten Aufwand für das Projektteam führen, wobei dies nicht das Hauptproblem dieser Variante darstellt. Denn durch die umfangreiche anfangs mit Sicherheit unübersichtliche Beispiel-DLL würden programmiertechnisch unerfahrene Benutzer schnell zurückschrecken und lieber Konkurrenzprodukte benutzen, als sich in dieses Produkt aufwendig einzulesen.

Durch diesen gesamten Analysevorgang kam das Projektteam zu dem Schluss, dass es für die zukünftige Zielgruppe des Produkts wohl am einfachsten und damit auch am sinnvollsten wäre, dieses Produkt mit einer GUI zu realisieren, da es sich bei dieser Zielgruppe meist zwar um Börsenerfahrene Personen handelt, diese aber wahrscheinlich nicht so standfest in der Welt des .net-Frameworks sind. Außerdem ermöglicht eine simple GUI, die Einbindung eines Algorithmus mittels Dateosystem-Browsing, ohne dass der Benutzer mühsamst den Pfad selbst suchen und in die Software einfügen muss. Als Variante zur Einbindung es algorithmus erscheint es dem Projektteam ebenfalls am sinnvollsten den Algorithmus einfach über ein Interface mit einer ganz normalen DLL einzubinden, da es aufwendiger und wahrscheinlich auch langsamer wäre, den Algrithmus als eingenständige Software über Sockets anzustprechen.

2.4 Wirtschaftliche Machbarkeit

2.4.1 Aufwandsabschätzung

2.4.2 Nutzen

2.4.3 Prüfen der Risiken

2.4.3.1 Personenausfall

Eintrittswahrscheinlichkeit: gering
Auswirkungen: hoch

In dem unerwarteten Fall, dass ein Teammitglied längerfristig ausfällt, muss es möglich sein die Arbeitsaufgaben dementsprechend neu aufteilen zu können. Folgende Fälle könnten auftreten:

- Streit im Team
- Ausfall durch Krankheit oder Tod eines Teammitglieds
- Austritt eines Teammitglieds aus dem Projekt
- Der Auftraggeber könnte aufgrund von Unklarheiten den Projektabbruch initiieren
- Es kann passieren, dass von Seite des Auftragsgebers plötzlich kein Interesse an der Umsetzung des Produktes mehr gegeben ist, und es dadurch zu extremem Zeitverzug kommt, was bis zum Abbruch führen kann
- Interessensverlusst eines Teammitglieds, und damit verbundene minderwertigere Arbeit

Folgende präventive Maßnahmen werden eingeführt:

- Regeln für den Umgang innerhalb des Projekts
- Ausreichendes Interesse jedes Mitglieds und keine leistungstechnische Probleme
- Gutes Verhältnis mit den Auftraggebern

2.4.3.2 Zeitliche Risiken

Eintrittswahrscheinlichkeit: gering
Auswirkungen: mittel

Die Aufwands- und Zeitschätzung basiert auf dem derzeitigen Lastenheft des Auftraggebers und stellt eine zeitgerechte Fertigstellung sicher. Sollten sich jedoch die Anforderungen des Kunden während des Projekts ändern, so wird sich das mit großer Wahrscheinlichkeit verzögernd auf den Fertigstellungstermin auswirken. Die mit dem Kunden vereinbarte Funktionsanalyse und die Meilensteine mit gemeinsam festgelegten Qualitätskriterien sollten jedoch diesem Risiko entgegenwirken.

2.4.3.3 Technische Risiken

Datenverlust
Eintrittswahrscheinlichkeit: gering
Auswirkungen: mittel

Aufgrund der nicht auszuschließenden Gefahr des Datenverlusts, muss dafür gesorgt werden die Sicherheit der Daten, sowie auch die Verfügbarkeit dieser zu garantieren. Dieses Problem wird mithilfe eines GIT-Server (GIT) gelöst, durch diesen Server ist es möglich die Versionen der Software immer zugänglich zu machen und zusätzlich die Daten auf den Computern der Projektmitgliedern zu speichern.

Hardwareausfall
Eintrittswahrscheinlichkeit: gering
Auswirkung: mittel

Es kann ohne jede Vorwarnung immer und überall ein Hardwareausfall passieren, dies kann selbst verschuldet, aber auch plötzlich passieren. Um mit einem solchen Ausfall zurecht zu kommen besitzt das Projektteam einen Ersatzlaptop um das gezielte Arbeiten auch nach dem Ausfall garantieren zu können.

Versionsverlust
Eintrittswahrscheinlichkeit: sehr gering
Auswirkung: mittel

Bei den Versuchen mit dem Algorithmus wird andauernd etwas in der Datei verändert. Bei dieser Tätigkeit kann es passieren das die Zusammenhänge zwischen 1 oder mehreren Versionen des Algorithmus verloren gehen. Bei so einem Verlust kann auch das grundliegende Verständnis für die jeweilige Version verloren gehen. Folgende präventive Maßnahmen werden eingeführt:

- Verwendung eines GIT, für die automatische Versionierung

- Zwingende Benutzung der Bugtrackingsoftware
- Kommunikation im Team über die Änderungen am Algorithmus

KAPITEL 3

The Proposed Architecture

"The market is not an invention of capitalism. It has existed for centuries. It is an invention of civilization."

(Mikhail Gorbachev)

3.1 Introduction

KAPITEL 4

Implementation

KAPITEL 5

Évaluation

"Measure what is measurable, and make measurable what is not."

(Gaileo Galilei)

KAPITEL 6

Conclusion and Outlook

"Success and failure are both greatly overrated. But failure gives you a whole lot more to talk about."

(Hildegard Knef)

ANHANG A

Appendix

Literaturverzeichnis

[1] G. Koppensteiner, R. Hametner, R. Paris, A. Passani, and M. Merdan, "Knowledge driven mobile robots applied in the disassembly domain," in *Automation, Robotics and Applications (ICARA), 2011 5th International Conference on*, dec. 2011, pp. 52 –56.

[2] M. Merdan, "Knowledge-based multi-agent architecture applied in the assembly domain," Ph.D. dissertation, Vienna University of Technology, 2009.

[3] F. L. Bellifemine, G. Caire, and D. Greenwood, *Developing Multi-Agent Systems with JADE V2*. Wiley, 2007. [Online]. Available: http://www.amazon.ca/Developing-Multi-Agent-Systems-Fabio-Bellifemine/dp/0470057475/sr=8-1/qid=1170365284/ref=sr_1_1/702-0885532-1303250?ie=UTF8&s=books

[4] Wikipedia contributors, "Wissenschaftliche arbeit," Sep. 2012, page Version ID: 107839675. [Online]. Available: http://de.wikipedia.org/w/index.php?title=Wissenschaftliche_Arbeit&oldid=107839675

[5] F. L. Bellifemine, G. Caire, and D. Greenwood, *Developing Multi-Agent Systems with JADE*. Wiley, 2007. [Online]. Available: http://www.amazon.ca/Developing-Multi-Agent-Systems-Fabio-Bellifemine/dp/0470057475/sr=8-1/qid=1170365284/ref=sr_1_1/702-0885532-1303250?ie=UTF8&s=books

Erklärung

Hiermit erklären wir, dass die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt wurde. Die aus anderen Quellen oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet.
Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder in ähnlicher Form in anderen Prüfungsverfahren vorgelegt.

Wien, im Oktober 2012

Name1

Name2

Name3

Name4