tgm
Die Schule der Technik

Machbarkeitsstudie

NOCTUA

Wertpapierhandelsalgorithmus mit Marktzustandsadaption

Ausgeführt in Zuge der Reife und Diplomprüfung
Ausbildungszweig Systemtechnik/Medientechnik

unter der Leitung von
Prof. Mag. Hans Brabenetz
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

Inhaltsverzeichnis

Abbildungsverzeichnis

Listings

0.1 Einleitung

Das algorithmische Handeln hat besonders in den letzten zwei Jahrzehnten rasant zugenommen. 2011 wurde davon ausgegangen, dass mindestens 30% des Aktienhandelsvolumen in den USA bereits algorithmischer Natur ist. Alleine schon die Kommissionen der Broker für algorithmisch abgewickelte Entscheidungen beziehen sich weltweit auf etwa $400-600 Millionen. Die meisten der verwendeten Algorithmen sind allerdings proprietärer Natur und dienen häufig nicht der direkten Optimierung des Profit, sondern dem Aufteilen großer Orders und der Minimierung des Risikos und der Kosten. [**?**]
Solche Algorithmen, die direkt Handelsentscheidungen treffen, existieren ebenfalls und werden häufig auch für Fonds eingesetzt, die an Privatpersonen weiterverkauft werden. Besonders kleinere Unternehmen und Privatpersonen haben nicht die Kapazitäten, um beim High Frequency Trading (HFT), wo jede Millisekunde zählen kann, mitzumischen. Klassische Handelssysteme können allerdings sehr wohl als Computersoftware umgesetzt werden und bieten die Vorteile schneller nach einem klar definierten System Entscheidungen zu treffen und dies gegebenenfalls auch ohne menschliche Aufsicht.
Märkte, und insbesondere Aktienmärkte, verhalten sich nicht immer während invariant, sondern können ihre intrinsischen Systematiken mit der Zeit ändern. Folglich kann es zu Einbußen bei der Performance von Algorithmen kommen, die ebendiese Systematiken ausnutzen. Um dies zu verhindern, oder zumindest die Effekte zu vermindern, muss ein erfolgreicher Algorithmus eine gewisse Adaption bzw. eine Anpassung und dadurch zusätzliche Flexibilität der Parameter ermöglichen. Ziel des Projektes ist es, genau solch ein System zu entwickeln und umzusetzen. Die vollständige Eigenentwicklung ermöglicht volle Einsicht in alle relevanten Bereiche des Algorithmus und der verwendeten Parameter. Außerdem kann mit Hilfe einer zu entwickelnden Backtesting-Software (BTS) gezielt die Performance des Algorithmus anhand diverser Kriterien gemessen werden. Im folgenden geht es nun darum, zu klären, welche Ansätze zur Umsetzung eines solchen Algorithmus möglich sind und wie die technische Implementierung dieser aussehen könnte. Zusätzlich soll die optimale Lösung, sowohl für alle Aspekte des Algorithmus, als auch der BTS erörtert werden.

0.2 Finanzwirtschaftliche Grundlagen

Aufgabe dieses Kapitels ist es zunächst ein Fundament für die Verständnis dieser Arbeit zu legen und Funktionsweisen, Vorgänge, sowie verschiedene Ansätze zu erläutern.

0.2.1 Börse

Eine Börse ist ein Handelsmarkt auf dem sich Preise durch Angebot und Nachfrage von Handelspartners bilden und Handel nicht direkt zwischen Käufern und Verkäufern, sondern über berechtigte Händler, abgewickelt wird. Wichtig ist dabei, dass immer für ausreichend Liquidität gesorgt werden muss, so dass jederzeit Wertpapiere gekauft und auch verkauft werden können.
Handelsvorgänge oder *Trades* resultieren aus einer Erwartungshaltung der Marktteilnehmer. [?] [?] Unter der Annahme von steigenden Kursen werden Einheiten gekauft, i.e. es wird *long* gegangen, werden fallende Kurse erwartet, werden entweder existierende Stückzahlen verkauft oder es wird sogar ein Leerverkauf getätigt, i.e. eine *short*-Position eröffnet. Damit wird der Verkauf von geborgten Wertpapieren bezeichnet, die zu einem späteren Zeitpunkt beim Schließen der short-Position erst gekauft werden. Sinken die Kurse dazwischen wird daher zu einem höheren Preis verkauft, als später gekauft wird und es entsteht die Differenz als Gewinn. [?]
Wollen mehr Handelsteilnehmer oder *Trader* kaufen, als verkaufen steigt der Preis aufgrund der hohen Nachfrage, man spricht auch von einem Bullenmarkt. [?] Ist es andersherum so, dass die Verkäufer überwiegen und der Preis sinkt handelt es sich um einen Bärenmarkt. [?]

0.2.1.1 Preise

Für jedes Wertpapier wird ein sogenanntes Orderbuch geführt, das die aktuellen Kauf- und Verkaufaufträge beinhaltet. Investoren interessieren sich meist für die sogenannte Quote-Zeile, die Informationen zu den günstigsten Konditionen sowohl auf Käufer-, als auch auf Verkäuferseite bietet.
Eine Quote-Zeile von Apple (AAPL) könnte dabei folgendermaßen aussehen.

Bid	Bid Size	Ask	Ask Size
691.52	700	691.66	300

Der Bid-Preis von $691.52 ist das höchste vorhandene Gebot für eine Apple-Aktie. Die Bid-Size gibt die Information über die Anzahl an Aktien, die Käufer zu diesem Preis erstehen wollen. Die Anzahl wird in *round lots* angegeben; meist

entspricht ein round lot 100 Stück der Aktie. Die Bid-Seite beschreit somit die Käuferseite.
Der Ask-Preis und die Ask-Size beschreibt hingegen das beste Angebot auf der Verkäuferseite. In diesem Fall werden 300 round lots für den Stückpreis von $691.66 angeboten.
Soll ein Handel so schnell als möglich abgewickelt werden muss zum Ask-Preis gekauft und zum Bid-Preis verkauft werden. In diesem Fall verändert sich die Quote-Zeile so, dass der nächstbeste Auftrag angezeigt wird. Diese Hintereinanderreihung von Angeboten wird auch als Orderbuchtiefe bezeichnet.
Angenommen ein Käufer ist nicht bereit zum aktuellen Ask-Preis zu kaufen. Er will bessere Konditionen und schickt eine *Limit-Order*, d.h. zu gegebenem Preis oder besser [1], mit einem Limit, das zwischen Ask- und Bid-Preis liegt. Sein Kaufgebot ist damit höher als der zuvor höchste und wird daher sofort in der Quote-Zeile auf der Bid-Seite angezeigt.

Der Aktienkurs wird mithilfe dieser vorhandenen Orders so gebildet, dass stets der höchste Umsatz entsteht. [**?**]

0.2.2 Markt

Aktienpreise setzten sich vollständig durch Angebot und Nachfrage zusammen. Nachfrage entsteht wenn Handelsteilnehmer annehmen, dass die Preise steigen, Angebot durch die Erwartung von negativen Kursveräufen.
Diese Annahmen beruhen auf einer Vielzahl von Informationen, die zur Verfügung stehen. Somit haben diese Informationen einen direkten Einfluss auf die Aktienkurse. Veröffentlicht ein Unternehmen seine Bilanz und fallen darin die Gewinne höher aus als erwartet, wird die Aktie aufgrund dieser Information steigen. Die Preise reflektieren aber nicht nur realisierte Gewinne, sondern auch potentielle zukünftige. Wären in der Theorie alle möglichen Informationen allen Handelsteilnehmern bekannt, würde sich der Kurs nur mehr durch das Auftauchen von neuen Informationen verändern. In der Praxis verhält es sich anders, da weder alle Informationen vorhanden sind, noch Handelsteilnehmer absolut rational agieren.
Diese in den 1960ern entwickelte Theorie wird als *Efficient Market Hypothesis* bezeichnet und besagt kurzgefasst, dass Marktpreise alle verfügbaren Informationen vollständig widerspiegeln.
Außer der Schlussfolgerung, dass Nachrichten und Ereignisse für Preisentwicklungen höchst relevant sind, lässt sich ebenfalls folgern, dass positive Informationen, z.B. eine Wachstumsprognose, keinen Kursanstieg bedeuten muss, weil diese schon in dem aktuellen Marktpreis inbegriffen sein kann. [**?**]
Unter der Prämisse, dass die Efficient Market Hypothesis uneingeschränkte Gültigkeit besitzt, lässt sich außerdem deduzieren, dass Preise nicht vorhergesehen

[1] IB-Ordertypen siehe: http://www.interactivebrokers.com/de/p.php?f=orderTypes

werden können. Wäre dies nämlich der Fall und könnte jeder Markteilnehmer Preise beliebig genau vorhersagen, wäre niemand bereit unter dem vorhergesagten Preis zu verkaufen und niemand würde darüber Kaufen. Daraus resultiert eine sofortiger Sprung auf den vorhergesagten Preis. Da nur neue Informationen, die *noch nicht bekannt sind* das Preisniveau verändern, sind folglich auch die Preise noch nicht bekannt. []
Da ein Marktpreis bei erhöhter Nachfrage steigt, ist es für Aktienanleger sinnvoll Geld dort anzulegen, wo auch andere investieren. Gewinn liegt daher darin herauszufinden was die Mehrheit oder der Markt denkt und nicht in der persönlichen Einschätzung der Information. John Maynard Keynes beschreibt diese Theorie schon 1936 in seiner *General Theory*.
In dieser Theorie lässt sich eine Rückkopplung an Information erahnen. Nicht nur die ursprüngliche Information selbst spielt eine Rolle, sonder auch die Reaktion darauf, im Falle von Aktien also die Kursentwicklung. Dieser iterative Prozess benötigt aufgrund von menschlichem Handeln aber auch seine Zeit. Dadurch entstehen durch Kursschwankungen und die verursachten Reaktionen Trendbewegungen in den Märkten. Erst durch diesen, eigentlich psychologischen Effekt, funktionieren Charttechniken und technische Analysen.
Für jene, die sich zusätzlich noch mit makroökonomischen Zusammenhängen befassen besteht trotzdem weiterhin ein Vorteil. Marktrückkopplungen unterliegen einem „Herdentrieb“ und können zu Spekulationsblasen führen. Übergeordnete Vorgänge können somit frühzeitige Hinweise liefern und bei der korrekten Interpretation von den resultierenden Kursverläufen helfen. [**?**]

0.2.2.1 Marktzustände

Um die Performance von Trendfolgemethoden zu optimieren sollen verschiedene *Marktzustände* unterschieden werden, die sich auf den Kursverlauf auswirken. Zu diesem Zweck können diverse Kriterien untersucht werden. Angefangen bei längeren Trendverläufen, die nicht zum aktiven Handel genutzt werden, über Währungswechselkurse bis zu Zinssätzen.
Beispielsweise könnte sich herausstellen, dass bei einem langfristigen Aufwärtstrend bei Kaufsignalen die long-Positionen immer größer ausfallen sollten, als die short-Positionen bei Verkaufssignalen oder dass erhöhte Volatilität und steigende Zinsen einen Kurseinbruch ankündigen.

0.2.3 Trends und Gleitende Durchschnitte

0.2.3.1 Trends

Bei der Trendbestimmung in Börsenkursen gelten gleitende Durchschnitte, oder Moving Averages (MAs), als ein beliebtes Mittel und infolge dessen häufig genutzter Indikator. Die klassische Technische Analyse ist oft subjektiv und unterliegt

einem gewissen Interpretationsspielraum, während MAs für algorithmische Handelssysteme klare Entscheidungen generieren können.
Ein MA ist wie das zweite Wort bereits verrät eine Art statistischer Durschnitt von einer Reihe von Werten. *Gleitend* oder *moving* bedeutet, dass nicht alle historischen Daten in die Durchschnittsberechnung einfließen, sondern nur eine beschränkte Anzahl der vergangenen Daten. Zur Berechnung von diversen MAs gibt es unterschiedliche Varianten zur Wahl der verwendeten Rohdaten. Klassischerweise werden Schlusskurse zur MA-Berechnung verwendet, wobei manche Trader es vorziehen, Open- oder Mittelwerte aus Open- und Close heranzuziehen. Eine weitere Variante ist eine separate Berechnung von MAs für High- und Low-Kurse, wodurch ein Band entsteht, das einen Art Neutralbereich anzeigt, der beispielsweise als Signalfilter verwendet werden kann.
Prinzipbedingt sind alle MAs Trendfolgeindikatoren, sie antizipieren keine zukünftigen Kursveränderungen, wie viele Methoden der Technischen Analyse, sondern reagieren nur auf bereits stattgefundene Trendwenden.

0.2.3.2 Simple Moving Average

Bei einem Simple Moving Average (SMA) wird für jeden neuen Wert jeweils aus den letzten n Werten ein arithmetischer Mittelwert berechnet. Soll aus einer Reihe von Close-Kursen ein 10-Tages-SMA berechnet werden, werden dazu die letzten 10 Schlusskurse addiert und anschließend durch 10 dividiert um einen Wert zu erhalten. Für den folgenden Mittelwert wird der älteste subtrahiert, ein weiterer neuer Wert addiert und die Summe anschließend wieder für einen Mittelwert durch 10 dividiert. Auf diese Weise entsteht ein geglätteter Kurs, der kurzzeitige Preisveränderungen abschwächt, wodurch Trends klarer erkennbar werden.
2 häufig kritisierte Eigenschaften des SMA sind, dass erstens nicht alle vorhandenen Daten in die Berechnung miteinbezogen werden und zweitens alle verwendeten Kursdaten mit gleicher Gewichtung in das Resultat eingehen. Bei einem 10-Tage-SMA hat jeder Kurswert, unabhängig vom Alter, eine Gewichtung von 10%, bei einem 20-Tage-SMA 5%.

$$P_t^* = \frac{1}{n} * \sum_{i=0}^{n} x_{t-n} \tag{1}$$

0.2.3.3 Linear Weighted Moving Average

Um eines der Probleme des SMA zu vermeiden, und zwar die gleiche Gewichtung von allen Daten über die Zeit, kann ein Linear Weighted Moving Average (LWMA) verwendet werden. Dabei werden Daten zu früheren Zeitpunkten geringer gewichtet, als aktuellere Daten. Konkret wird der neueste Eintrag mit n multipliziert, der davor mit $n-1$ und so weiter bis der letzte den Multiplikationsfaktor

1 erhält. Die daraus gebildete Summe muss anschließend noch durch die Summe der Multiplikatoren dividiert werden, um einen Durchschnitt zu erhalten. Bei einem 10-Tage-LWMA werden die Kursdaten von neu nach alt mit $10, 9, 8, 7, \ldots, 1$ multipliziert und die anschließend gebildete Summe durch $10+9+8+7+\cdots+1$ dividiert.

$$P_t^* = \frac{\sum_{i=1}^{n} P_i * i}{\sum_{i=1}^{n} i} \tag{2}$$

0.2.3.4 Exponential Moving Average

Der Exponential Moving Average (EMA) versucht beide Hauptkritikpunkte am SMA zu lösen, indem er neuere Werte immer stärker gewichtet und alle zur Verfügung stehenden Daten in die Berechnung miteinbezieht. Man spricht auch vom exponentiell geglätteten gleitenden Durchschnitt. Dazu muss ein *Smoothing Factor (SF)*, auch Glättungsfaktor genannt, zwischen 0 und 1 gewählt werden mit dem der neueste Wert multipliziert wird. Der zuvorige Wert wird mit der Differenz des SF und 1 multipliziert. Der aktuelle EMA ergibt sich dann aus der Addition dieser beiden gewichteten Werte. Dadurch bleibt jeder Kurswert unendlich lange in der zukünftigen Berechnung bestehen, dessen Bedeutung konvergiert aber gegen 0. Aus ebendiesem Grund handelt es sich mathematisch rigoros betrachtet bei dem EMA nicht wirklich um einen *moving* average, da sich der Wertebereich nicht verschiebt, sondern immer alle Werte verwendet werden. [**?**] Dies beschreibt eine rekursive Vorgangsweise; anders wäre es auch möglich mit dem ältesten (bzw. zweitältesten) Wert zu beginnen und die Berechnung bis zum neuesten fortzusetzen. Die folgende Formel soll die Berechnung des exponentiell geglätteten Preises P_t^* zum Zeitpunkt t verdeutlichen. Dabei ist α der SF und P_{t-1}^* der exponentiell geglättete Preis zum Zeitpunkt $t-1$.

$$P_t^* = \alpha * P_t + (1-\alpha) * P_{t-1}^* \tag{3}$$

[**?**]

Um den Lag, also das hinterherhinken hinter dem tatsächlichen Kurs zu reduzieren gibt es die Möglichkeit neue Daten noch stärker zu gewichten, als bei einem normalen EMA. 1994 von Patrick Mulloy eingeführt, schaffen sowohl der Double Exponential Moving Average (DEMA) als auch der noch stärker neue Daten gewichtende Triple Exponential Moving Average (TEMA), bei dieser Problematik Abhilfe. Den DEMA darf man sich allerdings nicht als EMA eines bereits berechneten EMA vorstellen. Das würde zu einer ungewünschten starken Abwertung aktueller Daten führen, wie an den Gewichtungsgraphen in Abbildung **??** zu sehen ist. Tatsächlich wird der DEMA aus einer Zusammensetzung aus einem

einfachen und einem doppelten EMA berechnet, wodurch ein neuer MA entsteht mit weniger Lag als jede der Komponenten. [?]

$$DEMA = 2 * EMA - EMA(EMA) \tag{4}$$

$$TEMA = 3 * EMA - 3 * EMA(EMA) + EMA(EMA(EMA)) \tag{5}$$

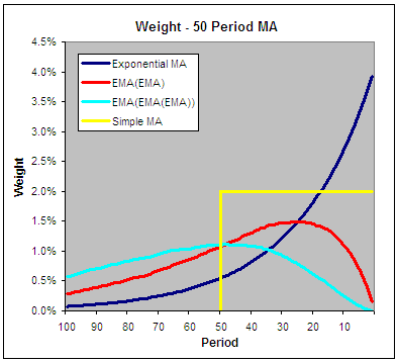

Abbildung 1: Falsche Double und Triple Exponential Average Gewichtung

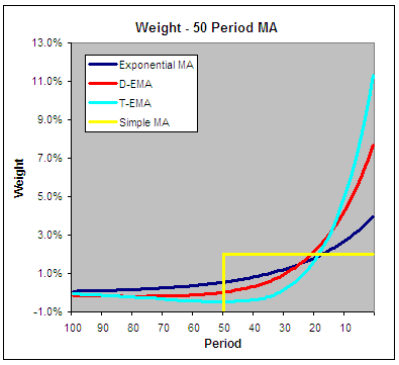

Abbildung 2: Richtige Double und Triple Exponential Average Gewichtung

[?]

0.2.3.5 Momentum

Das Momentum ist ein Oszillator, der die Kurssteigung über einen gewissen Zeitraum angibt. Aus der Differenz des Preises zum Zeitpunkt t und des Preises vor n Perioden ergibt sich eine einfach zu interpretierende Anzeige für die Stärke eines Trends. Aus einem 10-Tage-Momentum ist ablesbar, ob der Kurs aktuell höher notiert, als noch vor 10 Tagen. Wenn das Momentum von einer Position unter 0 bis zur Nullposition ansteigt ist ersichtlich, dass die Stärke eines aktueller Abwärtstrend abnimmt, wobei der Trend immer noch abwärts gerichtet sein kann. Er wird nur nicht mehr stärker. Erst bei einem positiven Wert des Momentum ist ein Aufwärtstrend zu erkennen.
Der große Vorteil dieses Indikators liegt in der Frühzeitigkeit seiner Anzeige. Bevor sich der Trend umkehrt, zeigt der Momentum-Oszillator bereits an, dass eine Trendwechsel bevorstehen könnte, aufgrund der abnehmenden Stärke des aktuellen Trends.

$$M_t = P_t - P_x \tag{6}$$

0.2.3.6 Relative Strength Index

Der Relative Strength Indicator (RSI) gibt ähnliche Informationen, wie das Momentum, schafft aber Abhilfe bei einigen Schwachstellen. Beispielsweise gibt es beim Momentum keine maximale Unter- und Obergrenze, wodurch es schwierig ist festzustellen, ob es sich momentan um eine Extremsituation handelt oder nicht. Der RSI bewegt sich hingegen immer zwischen 0 und 100.
Eine wesentlich schwerwiegendere Schwachstelle des Momentums ist aber seine absolute Abhängigkeit von dem Kurs vor genau n Zeitperioden. Sollte der Kurs zu der Zeit ein Hoch oder Tief erreicht haben, hat dies Auswirkungen auf das aktuelle Momentum, selbst wenn sich der aktuelle Kurs gar nicht oder nur kaum verändert.

$$RSI_t = 100 - \frac{1}{1 + RS_t} \tag{7}$$

RS_t ist in der Formel 7 die relative Stärke (relative strength) und wird berechnet aus dem Durchschnitt der Schlusskurse von n Tagen mit steigenden Kursen, dividiert durch den Durchschnitt der Schlusskurse von n Tagen mit fallenden Kursen. Der Erfinder sah ursprünglich eine Zeitperiode von $n = 14Tage$ vor, wobei aber für unterschiedliche Volatilität auch andere Parameter gewählt werden können. Interessant ist der RSI, wenn er im Vorhinein festgelegt Level überschreitet. Diese sind standardmäßig bei 70 und 30 für Überkauft- und Überverkauft-Level, können aber beispielsweise als Anpassung für Bullenmärkte auf 80 oder für Bärenmärkte auf 20 verändert werden.

0.2.3.7 Aroon

Der 1995 von Tushar S. Chande entwickelte Aroon ist ein zweigeteilter Trendindikator, der zwar nicht die Stärke eines Trends anzeigt, aber gut die Marktlage erfasst. So kann beispielsweise erkannt werden, ob es sich um eine Trend- oder eher um eine Seitwärtsphase handelt. Die Signale sind dabei sehr einfach abzulesen und fluktuieren sehr wenig, da die Trendstärke, wie bereits erwähnt, nicht direkt erfasst wird, sondern nur zeitlich, sodass ein Verlust des Momentums feststellbar wird. Die Stärke des Indikators ist es, schnell einen Trendwechsel bzw. Marktzustandswandel anzuzeigen. Die klaren Signale eignen sich außerdem gut für algorithmische Handelssysteme, wenn die Stärke des Trends durch eine andere Komponente erfasst wird.
Aroon besteht aus einem Aroon-Up und einem Aroon-Down, die jeweils die vergange Zeit zu einem Kursextrem anzeigen und zwischen 0 und 100 variieren. Bei einem 14-Tage-Aroon hat der Aroon-Up bei einem aktuellen 14-Tages-Hoch einen Wert von 100.
Zur Interpretation ist eine Signallinie sehr wichtig, die von Chande bei einem Wert von 75 angesetzt wurde, aus anderen Quellen sind auch Werte um die 90

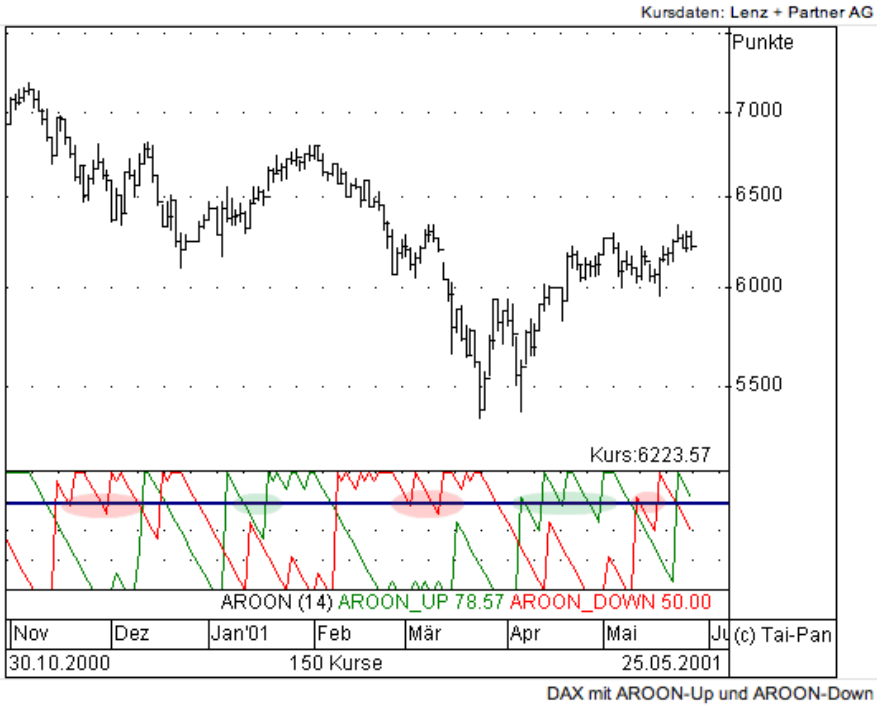

Abbildung 3: 14-Tage Aroon über 7 Monate des DAX-Kurses

zu finden. Liegt der Aroon-Up über dem Aroon-Down und steigt zusätzlich noch über die Signallinie ist dies als Aufwärtstrend zu identifizieren. Da der Indikator häufig kurzfristig unter die Signallinie fällt (s. 3), ist es noch nicht ratsam zu diesem Zeitpunkt die Positionen zu liquidieren, sondern noch zu warten bis sich die Up- und Down-Linien ebenfalls überkreuzen.
Ein Seitwärtstrend wird angezeigt, wenn sich Aroon-Up und -Down unter der Signallinie befinden, was bei liquiden Positionen nur kurzfristig auftritt. [**?**]

Die Berechnung des Aroon-Up und Aroon-Down ist mit den folgenden Formeln beschrieben. Wobei $high_{n+1}$ in 8 die Zeit seit dem höchstem Hoch innerhalb von $n+1$ Perioden und low_{n+1} in 9 die Zeit seit dem tiefstem Tief innerhalb von $n+1$ Perioden bezeichnet.

$$\frac{n - high_{n+1}}{n} * 100 \tag{8}$$

$$\frac{n - low_{n+1}}{n} * 100 \tag{9}$$

0.2.3.8 Commodity Channel Index

Entwickelt wurde der Commodity Channel Index (CCI), wie der Name bereits andeutet, für Commodities, also Handelsgüter, kann aber ebenso gut für Wertpapiere und Derivate verwendet werden. Im Prinzip ist er ein Trendfolger, der sowohl Momentum, als auch Preis berücksichtigt. Der CCI zeigt den Handelspreis im Verhältnis zu einem MA über eine bestimmte Zeitdauer und wurde vom Erfinder Donald R. Lambert mithilfe eines Divisors, der auf der Standardabweichung basiert, normalisiert. Daher bewegt sich der CCI meist zwischen den Werten -100 und +100. Wenn sich der Preis stärker verändert als es im Zeitraum des MA bereits passiert ist, bewegt sich der CCI über diese Grenzen hinweg. Vom Erfinder wurde eine long-Position bei Werten über +100 und eine short-Position unter -100 empfohlen. Heutzutage benutzen viele Trader diese Grenzen in umgekehrter Manier als Support- und Resistance-Level, wodurch sie bevor sich ein Trend umkehrt auf einen möglichen Umschwung hingewiesen werden. Solange sich der CCI zwischen den beiden signifikanten Werten bewegt spricht man von einem trendlosen Markt.
Berechnet wird der CCI, indem vom typischen Preis (s. 10) der SMA abgezogen wird und die Differenz, wie in Formel 11 beschrieben, durch die Standardabweichung dividiert wird. Der Gewichtungsfaktor von $\frac{1}{0.015}$ wird gewählt damit die Hauptzahl der Werte zwischen -100 und 100 liegt. Zeigt der CCI in eine inverse Richtung zum MA, deutet diese Divergenz auf einen instabilen Markt. Es könnte ein Trendwechsel bevorstehen.

$$TP_t = \frac{H + L + C}{3} \tag{10}$$

$$CCI = \frac{TP_t - SMA(TP_t)}{\sigma(TP)} * \frac{1}{0.015} \tag{11}$$

[?]

0.2.4 Unterstützung und Widerstand

Die Kurse von Aktien unterliegen, wie bereits erwähnt gewissen Trends, die entweder auf-, ab- oder auch seitwärts verlaufen, wenn weder das eine, noch das andere klar feststellbar ist. Bei steigenden Kursen erreichen diese irgendwann ein Level, bei dem der Preis nicht mehr als billig angesehen wird und der Kurs dreht um. Der Trend verliert also seine Nachhaltigkeit. Dieses Level wird als *Widerstand* oder *Resistance* bezeichnet, da dieser nur schwer überwunden wird. Sinkt der Kurs eine Weile wieder und sonstige Bewertungskriterien des Handelsgutes haben sich nicht signifikant verändert, erreicht der Kurs einen Punkt, bei dem er wieder interessant für etwaige Käufer wird. Diese Schwelle wird als *Unterstützung* oder *Support* bezeichnet. Am Beispiel von Microsoft kann diese Funktion in der

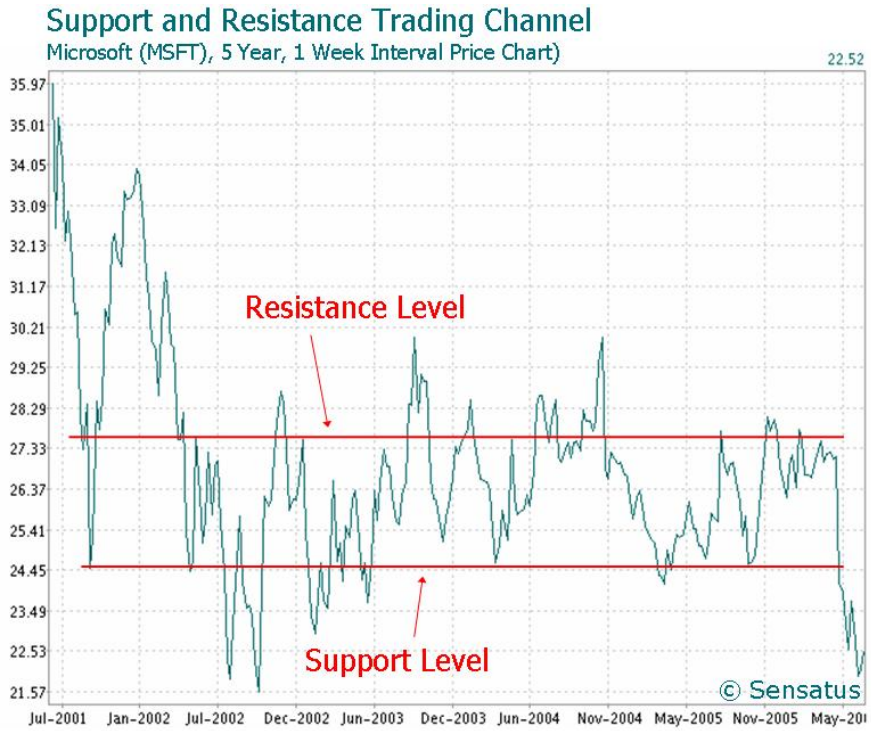

Abbildung 4: Microsoft 5-Jahres-Chart mit 1-Wochen-Intervallen und eingezeichneten Support- und Resistance-Level

Grafik 4 über einen längeren Zeitraum beobachtet werden.
Diese beiden Schwellen sind temporäre Hilfsmittel und lassen Wendepunkte erahnen. Bei Durchbruch eines Levels, kann hingegen mit einer Fortsetzung dieses Trends gerechnet werden und ggf. wird ein alter Widerstand zur Unterstützung. [?] Das Chart 5 von Paychex Inc. veranschaulicht diesen Wechsel von Unterstützungs- zu Widerstands- und wieder zurück zu Unterstützungslevel.

0.2.4.1 Prozentbänder

Prozentbänder sind eine andere Möglichkeit Nachhaltigkeit bzw. Überkauft- oder Überverkauft-Level festzustellen. Dazu wird ein MA um einen festgelegten Prozentwert nach unten und oben verschoben, so dass ein Kanal bzw. ein Band entsteht, in dem der Kurs die meiste Zeit verläuft. Wenn der Kurs über dem oberen Band notiert, wird dieser meist als überkauft angesehen, da er im Vergleich zum Durchschnitt sehr schnell gestiegen ist. Um wie viel Prozent die Bänder nach oben und unten verschoben werden, hängt von der Länge des MA und somit dem Tradingzeitraum ab. Oft gebraucht werden 3%-Bänder und eine 21-Tage-Linie oder 10%-Bänder mit einem 40-Wochen-MA.

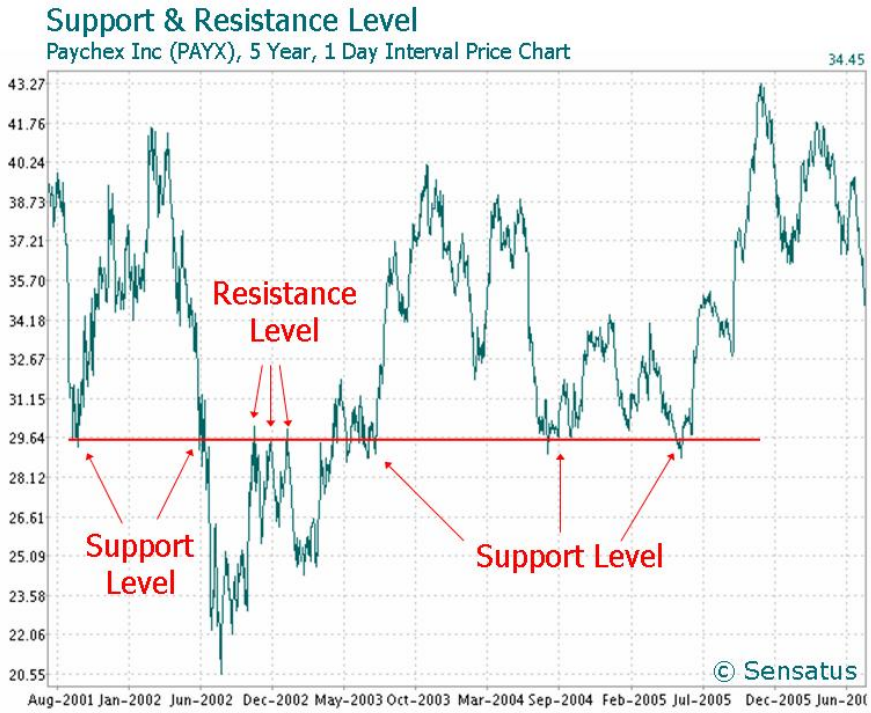

Abbildung 5: Paychex Inc. 5-Jahres-Chart mit 1-Wochen-Intervallen; Level wechselt zwischen Support- und Resistance-Funktion

0.2.4.2 Bollinger Bänder

Die Bollinger-Bänder, entwickelt von John Bollinger, sind ein ähnlicher Indikator, wie die Prozentbänder. Es wird ebenso ein Bereich oberhalb und unterhalb des MA aufgespannt, aber anstatt ihn einfach um einen fixen Prozentwert zu verschieben, wird die Durchschnittslinie um die *Doppelte Standardabweichung*, die meist auf Länge des MA berechnet wird, nach oben und nach unten verschoben. Dabei dienen die Bänder meist als Kursziele, so dass beim ansteigen des Kurses vom unteren Bollinger-Band, das obere als Ziel angenommen wird. Der Durchmesser der Bänder zeigt die Volatilität des Kurses an, da bei geringer Volatilität die Standardabweichung ebenfalls sinkt, ergo werden die Bänder schmäler.

0.2.5 Handelsstrategien

In diesem Punkt sollen verbreitete Handelsstrategien erläutert werden, um einerseits potentiell zu testende Algorithmen festzustellen und andererseits eine Grundlage für Weiterentwicklung und Kombinationsmöglichkeiten aufzuzeigen.

0.2.5.1 Preiskreuzung des Gleitenden Durchschnitts

Eine simples Tradingmodell, dass von einigen Tradern verwendet wird, basiert auf einer einfachen Überkreuzung eines MA über den Preis. Es wird dabei angenommen, dass wenn der Preis über den MA steigt, ein Aufwärtstrend eingesetzt hat, da der Kurs begonnen hat schneller als der längere Durchschnitt zu steigen. Umgekehrt wird angenommen, dass bei einem Abfall des Kurses unter den MA ein Abwärtstrend folg und daher wird ein short-Signal generiert. Eine zusätzliche Sicherheit ist gegeben, wenn der MA selbst in erwartete Kursrichtung dreht.
Dieser simple Algorithmus hat einige Nachteile. Es ist zu jedem Zeitpunkt ein Signal gegeben, was bedeutet dass immer entweder eine long- oder eine short-Position gehalten wird. Außerdem verliert diese Strategie bei langen MAs nach der Trendumkehr wieder viel des Gewinns, da das Gegensignal erst spät generiert wird. Kurze MAs erzeugen oft Fehlsignale.

0.2.5.2 Verwendung Mehrerer Gleitender Durchschnitte

Eine häufig verwendete Variante zur Signalgenerierung mit MAs wird *Double Crossover Method* genannt. Dabei kommen 2 unterschiedlich lange MAs zum Einsatz, wobei ein Signal erzeugt wird wenn sich beide schneiden. Kreuzt der kurze MA den längeren entsteht ein Kaufsignal, auch *Golden Cross* genannt, und vice versa. Diese Variante erzeugt weniger Fehlssignale, als die direkte Verwendung des Preises, hinkt dem Markt dafür aber auch stärker hinterher. Die Länge der Durchschnitte hängt wie immer sowohl vom Handelszeitraum und der gewünschten Signalanzahl, als auch vom Markt ab.

Dieses System kann noch um einen weiteren MA raffiniert werden. Der Einsatz dreier Durchschnitte, oder *Triple Crossover Method* verfeinert die Art des Signals. Ein vollständiges Kaufsignal entsteht sobald der kurze über dem mittleren, und jener wiederum über dem langen MA notiert. Eine umgekehrte Anordung ist als Verkaufssignal anzusehen.
Beginnt der kurze Trendfolger über den mittleren und anschließend auch über den langen zu steigen, ist dies als vorbereitendes Kaufsignal zu interpretieren. Wenn der mittlere anschließend den langen auch noch von unten schneidet ist das Signal vollständig.
Auf diese Art kann beispielsweise bei unklaren Signalen eine Neutralstellung (i.e. keine Aktien im Portfolio) eingenommen werden oder die Market-Exposure (i.e. Anzahl der Aktien) reduziert werden.
Normalerweise werden für solche Systeme SMAs verwendet, wobei aber besonders bei einem Double Crossover System die Anwendung eines EMA oder sogar

0.2.5.3 Moving Average Convergence/Divergence

Der Moving Average Convergence/Divergence (MACD) ist ein Indikator, um die Differenz zwischen zwei EMAs darzustellen. Der längere EMA wird vom kürzeren abgezogen, um bei einem Kreuzungspunkt genau einen Wert von 0 zu erhalten. Eine andere Möglichkeit eines MACD-basierten Handelssystems ist es den MACD mit einem zusätzlichen EMA, der kürzer ist als beide zur Berechnung des MACD verwendeten EMAs, zu vergleichen. Dies wird als Signallinie bezeichnet. Obwohl die Signallinie den kürzesten Zeitraum abbildet, ist der MACD näher am tatsächlichen Kurs. Wenn der MACD die Signallinie von unten schneidet, entspricht dies daher einem Kaufsignal und umgekehrt.

0.3 Voruntersuchung

0.3.1 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 MACD und der CCI. Die meisten Algorithmen benutzen außerdem eine Zusammensetzung aus verschiedenen 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 [2] 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.

0.3.2 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 versuchen lediglich durch Korruption der Konkurrenz, selbst den wirtschaftlich

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

Abbildung 6: MetaTrader 5 Oberfläche

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.

0.3.2.1 MetaTrader 5

Hierbei[3] handelt es sich um ein ziemlich umfangreiches, ein wenig älteres Produkt, dass Futures, Options und Aktiendaten anbietet, aber eigentlich auf Foreign Exchange Market (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. Dieses wurde mit der Version 5 auch noch weiter vergrößert, was viele langjährige Nutzer ausdrücklich loben. Die Oberfläche des MetaTraders sieht in etwa aus, wie in Abbildung 6 dargestellt.
Doch das größte Feature des MetaTraders ist die eingebaute IDE (siehe Abbildung 7), 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

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

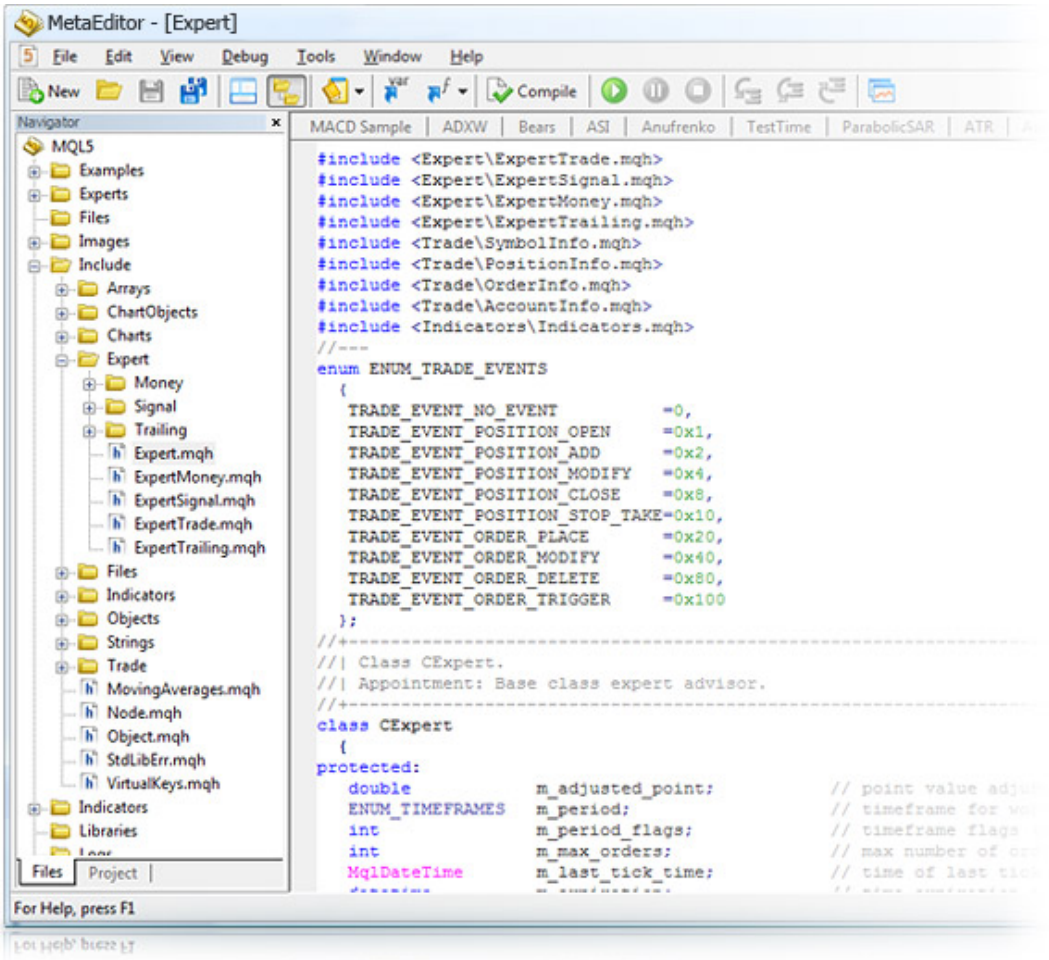

Abbildung 7: MetaTrader 5 IDE-Oberfläche

weiter gegenüber der MQL4 verbessert. Dies führt uns allerdings auch 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.

0.3.2.2 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.

0.3.3 SOLL-Zustand

Man muss also mehrere Problemstellungen ins Auge fassen, es gibt viele Algorithmen, allerdings wenige die zugänglich sind. Weil man also die nicht veröffentlichten nicht kennt, ist es schwer, Aussagen darüber zu treffen wie performant diese sind. Allerdings gibt es auch bereits Software die jedem ermöglicht den eigenen Algorithmus zu testen, dies jedoch nicht in kurzer Zeit und mit einer Analyse. Außerdem müsste man dazu eine neue Programmiersprache erlernen. Natürlich kann man nicht sagen, dass die ëineP̈rogrammiersprache jedem bekannt ist, aber es ist möglich sagen zu können das C, Java oder eine .net-Sprache bekannter ist als eine Software eigene Programmiersprache namens MQL5. Mit einer BTS die nur mit historischen Daten funktioniert kann man logischerweise den Vergleich zu anderen Algorithmen in dem gleichen Zeitraum klarer erkennen, aber auch die Einwirkung der verschiedenen Marktzustand die zu verschiedenen Zeitpunkten in den Daten stattgefunden haben identifizieren und vergleichen. Mit dieser Möglichkeit und Art Algorithmen zu vergleichen und zu testen, soll es dem Projektteam möglich gemacht werden eine Reihe an Algorithmen auszuwählen, die zu gewissen Marktzustandn verschiedene Performances gezeigt haben zu implementieren und durch die frühe Zustandserkennung eine Möglichkeit zu entwickeln zwischen den Algorithmen zu wechseln. Um dies realisieren zu können ist es notwendig das Risiko des getesteten Algorithmus zu bestimmen und zu beschreiben. Zu diesen Beschreibungen muss man dann auch eine Performancemessung (e.g. sharpe ratio) angeben, die je nach Marktzustand gegliedert ist, anführen. Eine Schwierigkeit dabei ist die Bestimmung der Zeit einer Klassifizierung, also wie lange zum Beispiel die Klassifizierung als Seitwärtstrend zutrifft und damit gerechnet wird.

0.3.3.1 Möglichst geringes Risiko

Durch die Entwicklung eines eigenen Algorithmus soll das Risiko das einem Aktionär zu Grunde liegt, möglichst gering gehalten werden. Deswegen ist es notwendig einen Algorithmus mit Werten einer BTS zu unterlegen. Dies sichert nicht nur den möglichen Verkauf oder die Funktionalität des Algorithmus, es erzeugt auch eine Vertrautheit mit der Funktion, die sonst nur sehr Wage erhofft werden kann. Im Prinzip entsteht also eine Art Möglichkeit unterschiedliche Algorithmen zu unterschiedlichen Marktzuständen miteinander zu vergleichen und deren Performance zu messen.

0.3.3.2 Weiterbildung der Projektteammitglieder

Im Laufe des Projekts wird die Teamfähigkeit und soziale Kompetenz aller Projektteammitglieder gestärkt. Das Projektteam ist an einer großen Erfahrungssammlung im Bereich der Finanzwirtschaft interessiert und wird diese auch während der Projektdurchführung gewinnen. Außerdem werden ihnen das Konzept

der Börse und die damit verbundenen Algorithmen näher gebracht. Es ist außerdem nicht zu übersehen, dass bei der erfolgreichen Entwicklung eines sich selbst regelnden Algorithmus, der Eigengebrauch für die Projektteammitglieder nicht untersagt sein wird.

0.4 Produktfunktionen

0.4.1 Must-Have

/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	-

0.5 Technische Machbarkeit

0.5.1 Variantenbildung

0.5.1.1 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	81	2	38	1	43
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	81	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.

Nach dem Festlegen der verwendeten Programmiersprachen ist es nun auch möglich in C# zwischen zwei verschiedenen Grafikframeworks für die GUI der BTS auszuwählen, weil Windows Presentation Foundation (WPF) neuer ist und Windows Forms (WF) nicht auf dem Model-View-Control-Pattern (MVC) basiert, wird das Framework WPF angewendet.

0.5.1.2 Backtesting-Software

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 drei 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?
- In welcher Weise werden die Algorithmen, die von der BTS getestet werden, versioniert und wie werden die Testergebnisse gespeichert?

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.

Da es sich bei Noctua um ein Projekt handelt, welches versucht einen möglichst profitablen Handelsalgorithmus zu erstellen und die BTS eigentlich nur ein Mittel zum Vergleich der unterschiedlichen Algorithmen ist, liegt es nahe, dass im Zuge der Algorithmuserforschung alle geschriebenen Algorithmen zu jedem Testzeitpunkt gespeichert werden sollten. Außerdem wäre es erheblich von Vorteil wenn man eine Möglichkeit hätte immer alle Versionen des Algorithmus anhand ihrer von der BTS ermittelten Testergebnisse vergleichen können. Dafür muss eine Ansammlung an Daten erzeugt werden, die es ermöglicht auch auf Rückschritte in der Performance des Algorithmus aufmerksam zu werden. Zudem sollen diese Daten, zur Senkung des Aufwandes beim Tester, automatisch von der BTS erstellt werden. Dafür gäbe es nun zwei Möglichkeiten. Man könnte eine Datenbank aufbauen, die jeweils die getestete Version des Algorithmus zusammen mit

seinen Testergebnissen (Performance, ...) in einem Datensatz speichert. Diese Lösung klingt grunsätzlich recht sinnvoll, doch bringt sie zwei Probleme mit sich. Das erste Problem ist die Speicherung des Algporithmus. Dieses könnte man allerdings über die Speicherung des algorithmus.dll-Files als Binary-Largs-Object oder durch die Speicherung des Pfades zur Algorithmus-Datei lösen. Allerdings ist es nicht so einfach die generellen Userwünsche zu ändern. Es ist nämlich nur ein kleiner Anteil unserer zukünftigen User gewillt, sich eine Datenbank einzurichten, nur um eine Versionierung für ihre Algorithmen zu erhalten. Deswegen hat sich das Projektteam entschieden, von der BTS einfache Dateien generieren zu lassen, die den Pfad zur Algorithmus-Datei und die dazu ermittelten Testergebnisse speichert. Außerdem wird im Zuge der Umsetzung von Noctua eine Dateistruktur für sowohl die Dokumentations-, als auch die Algorithmus-Dateien entworfen.

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 der 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 Dateisystem-Browsing, ohne dass der Benutzer mühsamst den Pfad selbst suchen und in die Software einfügen muss. Als Variante zur Einbindung des 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 Algorithmus als eigenständige Software über Sockets anzusprechen. Zur Versionierung der Algorithem wird die BTS eine Dateistruktur aus Dokumentations- und Algorithmus-Dateien anlegen, die alle Testergebnisse und Algorithmen speichert.

0.6 Wirtschaftliche Machbarkeit

0.6.1 Aufwandsabschätzung

0.6.1.1 Personalaufwand

Kategorie	Anzahl	Klassifizierung	Gewichtung	Summe
Eingabedaten	0	einfach	2	0
	0	mittel	3	0
	1	komplex	5	5
Abfragen	0	einfach	2	0
	2	mittel	3	6
	2	komplex	5	10
Ausgabedaten	1	einfach	3	3
	0	mittel	4	0
	0	komplex	6	0
Datenbestände	1	einfach	6	6
	1	mittel	9	9
	3	komplex	14	42
Referenzdaten	0	einfach	4	0
	0	mittel	6	0
	0	komplex	9	0
Summe			**E1**	**81**

Einflussfaktoren (können den errechneten Function Point-Wert um +/- 30% ändern)			
	1	Verflechtungen mit anderen Anwendungssystemen (0.5)	0
	2	Dezentrale Daten, dezentrale Verarbeitung (0.5)	4
	3	Transaktionsrate (0-5)	0
	4	Verarbeitungslogik	
	a	Rechenoperationen (0-10)	9
	b	Kontrollverfahren (0-5)	5
	c	Ausnahmeregelungen (0-10)	6
	d	Logik (0-5)	5
	5	Wiederverwendbarkeit (0-5)	5
	6	Datenbestands-Konvertierungen (0.5)	4
	7	Anpassbarkeit (0-5)	3
Summe der 7 Einflussfaktoren		**E2**	**41**
Faktor Einflussbewertung E3 = (E2 / 100) + 0,7	**E3**	**1,11**	
Bewertete Funtion Points E1 x E3			**89,91**

Abbildung 8: Functions-Point-Analyse zu Noctua.

Aus der Abbildung 8 ist ersichtlich, dass die Function-Points-Analyse ca. 90 Punkte ergibt. Anhand der Tabelle aus Abbildung 9 ergibt sich für 90 Function-Points eine ungefähre Projektdauer von 3 Personenmonaten. Dies entspricht bei einer Umrechnung von 40 Stunden pro Woche eine Gesamtstundenanzahl von 480 Arbeitsstunden. Dies wiederum geteilt durch die Anzahl der Projektteammitglieder ergibt das eine Workload von ca. 160 Stunden pro Projektteammigtlied.

Bei einem Stundensatz von 50,00 € pro Arbeitsstunde ergibt dies ein benötigtes Kapital von 24.000,00 € für das gesamte Projekt Noctua.

Function Points	PM	Function Points	PM	Function Points	PM
150	5	500	33	850	61
200	9	550	37	900	65
250	13	600	41	950	70
300	17	650	45	1000	75
350	21	700	49	1050	84
400	25	750	53	1100	93
450	29	800	57	usw.	

Abbildung 9: IBM Umrechnung von Functions-Points in Personenmonate aufgrund von Erfahrungswerten.

0.6.1.2 Materialaufwand

Da es sich bei diesem Projekt um ein reines Softwareprojekt handelt, bezieht sich der Materialaufwand ausschließlich auf die Lizenzkosten für die verwendete Software. Verbrauchsmaterial kommt in diesem Projekt nicht zum Einsatz.

Die Lizenzenkosten belaufen sich auf folgendes:

- 3 * 1.528,00 € für eine Microsoft Visual Studio Professional Lizenz für jeden unserer Computer. Diese dient als Entwicklungsumgebung.
- 1 * ca. 191,00 € für eine Microsoft Visio Standard 2010, zur Modellierung von Diagrammen.
- 3 * ca. 92,00 € für eine Windows 7 Home Premium Lizenz für jeden unserer Computer.
- 1 * ca. 345,00 € für eine Microsoft Project Standard 2010 Lizenz, zur Planung des Projekts.
- 1 * ca. 65,00 € für eine 2-9 Personen Lizenz von SmartGit mit 1 Jahr Support. Dies wird als GIT-Server (GIT)-Client benutzt.

Dies ergibt einen Gesamtmaterialaufwand von ca. 5.461,00 €.

0.6.1.3 Investitionskapital

Der aller wichtigste Teil des Investitionskapitals ist ein Laptop für jedes der Projektteammitglieder, um die Programmierarbeit überhaupt zu ermöglichen. Die Abschreibung dafür beläuft sich laut der Abschreibung für Abnutzung (AFA) für Laptops ab 400,00 € auf ca. 1.500,00 €.

Zusätzlich fallen kosten für den Arbeitsraum an. Man kann für drei Personen von einem 30 Quadratmeter großen Büroraum inklusive WC-Einrichtungen ausgehen. Inklusive Heizung fallen hierfür ca. 13,00 €/m^2/Monat an. Das beläuft sich dann auf ca. 4.680,00 €/Jahr. Zusätzlich kann man ca. 500,00 €/Jahr für Strom und Wasser annehmen.

Dadurch beläuft sich das gesamte benötigte Investitionskapital auf ca. 6.680,00 € für die gesamte Projektdauer.

0.6.2 Nutzen

0.6.3 Prüfen der Risiken

0.6.3.1 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

0.6.3.2 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.

0.6.3.3 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 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

0.7 Projektorganisation

Der Grund dafür, dass dieses Projekt ins Leben gerufen wurde ist das grundlegende Interesse und der Wissensdurst an der Finanzwirtschaft, sowie die Projektauftraggeber sind Herr Professor Mag. Hans Brabenetz und Herr Professor Dr. Helmut Vana. Der Projektleiter ist Peer Nagy und das Team besteht noch zusätzlich aus zwei weiteren Entwicklern. Die anderen Projektteammitglieder sind Herr Gabriel Pawlowsky und Herr Josef Sochovsky.

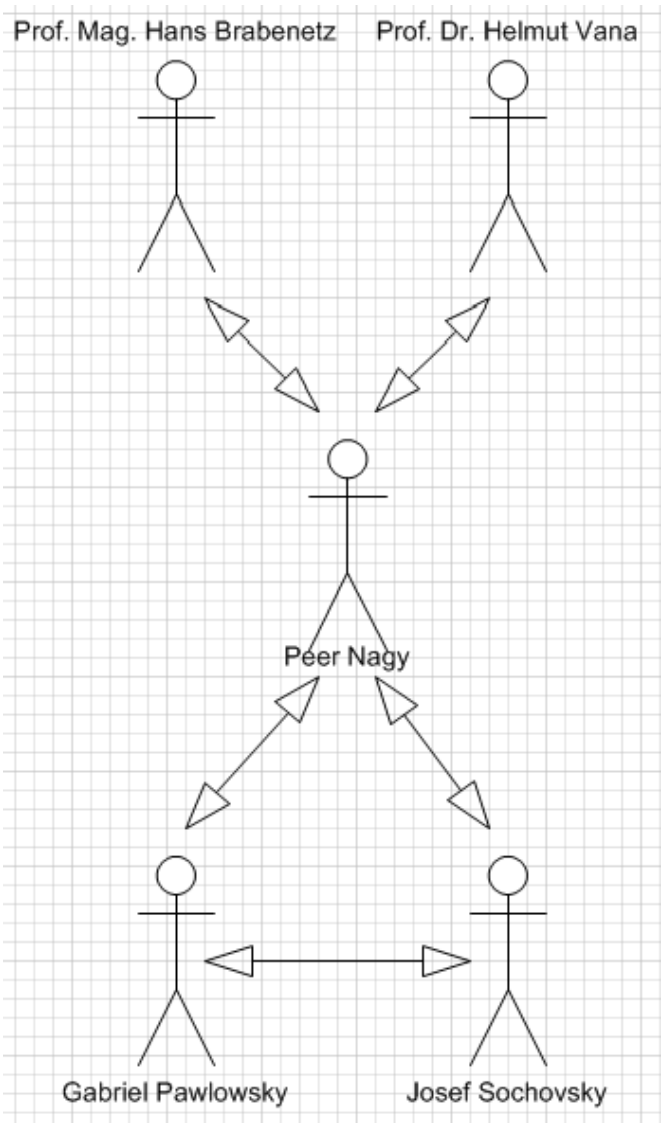

Abbildung 10: Projektorganisationsdiagramm erstellt mit Microsoft Visio

ANHANG A

Appendix

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