tgm
Die Schule der Technik

Machbarkeitsstudie

AQUILA

Tradingsoftware mit Controlling-Website

Ausgeführt in Zuge des Projektmanagement-Unterrichts im 5. Jahrgang
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

Version	**Autor**	**QS**	**Datum**	**Status**	**Kommentar**
0.1	Pawlowsky	Sochovsky	22.09.2012	draft	Erstversion
0.2	Sochovsky	Pawlowsky	26.09.2012	draft	Technische Machbarkeit
0.3	Pawlowsky	Nagy	27.09.2012	draft	Produkt-funktionen
0.4	Nagy	Sochovsky	27.09.2012	draft	IST-Erhebung und Zustand
0.5	Pawlowsky	Sochovsky	28.09.2012	draft	FPA
0.6	Nagy	Pawlowsky	29.09.2012	draft	Kosten/Nutzen
0.7	Sochovsky	Pawlowsky	06.10.2012	draft	Projekt-organisation
0.8	Pawlowsky	Nagy	07.10.2012	draft	Aufwands-abschätzung
0.9	Sochovsky	Nagy	10.10.2012	draft	Sollzustand
1.0	Nagy	Sochovsky	17.10.2012	draft	Management-Summary, Einleitung
1.1	Pawlowsky	Nagy	21.11.2012	final	Letzte Fehler ausgebessert

Inhaltsverzeichnis

Abbildungsverzeichnis

KAPITEL 1

Einleitung

Aktienhandel ist meist mit viel Erfahrung verbunden. Je kurzfristiger gehandelt wird, desto mehr Konzentration und Aufmerksamkeit muss den Vorgängen gewidmet werden, damit auch bei mehreren Trades am Tag summa summarum eine positive Bilanz entsteht. Dies macht es für kleine Firmen schwierig und für handelnde Privatpersonen nahezu unmöglich in diesem Zeitraum zu operieren. Das Ziel dieses Projektes ist es, genau bei dieser Zielgruppe zu punkten, indem eine Software zur algorithmischen Abbildung eines Handelssystems geschaffen wird, die automatisch Kauf- und Verkaufentscheidungen trifft. Eine Website als Schnittstelle zum Benutzer gewährleistet Plattformunabhängigkeit, gibt Informationen über Performance und Preisentwicklungen und ermöglicht das Ändern von Parametern von überall. Zusätzlich können mehrere Benutzer von unterschiedlichen Standorten mit einem gemeinsamen Portfolio operieren.
Charts auf der Website bieten live einen Überblick über die aktuelle Situation und dadurch eine erhöhte Transparenz der Arbeitsweise und Entscheidungsgenerierung des Algorithmus. Relevante Parameter, wie die zu handelnden Aktien oder die Höhe der Investition, können ebenfalls einfach über die Webschnittstelle verändert werden.

KAPITEL 2

Voruntersuchung

2.1 IST-Erhebung

Der Wertpapierhandel erfährt eine zunehmende Professionalisierung, was dazu führt, dass sowohl semiprofessionelle, als auch professionelle Trader in großen Unternehmen mit immer größeren Datenmengen, mehr Informationen und durch schnellere Märkte gleichzeitig mit kürzerer Entscheidungsdauer konfrontiert werden.
Um dem Trend zu programmatischen Trading-Lösungen entgegenzukommen gibt es eine Reihe von Datenanbietern, die Kursdaten in unterschiedlichen Zeitabständen und mit unterschiedlicher Aktualität anbieten. Je aktueller und öfter man die Ticks, also Kursstände zu einem Zeitpunkt bekommt, desto teurer ist die Datenanbindung im Durchschnitt. Einer der führenden Anbieter von Echtzeit-Marktdaten ist eSignal[1], der eine Selektion an Produkten zu verschiedenen Konditionen anbietet.[2] Die Version der Software „eSignal OnDemand“ bietet die Möglichkeit Intraday-Kursdaten zu günstigeren Konditionen 15 Minuten verzögert abzurufen. Für einen Produktivbetrieb ist diese Verzögerung vermutlich zu lange, aber sicherlich hinreichend für den Entwicklungs- und Testbetrieb.
Die Anbindung des Datenproviders kann über Dynamic Data Exchange (DDE) erfolgen.

Um eine individuelle Trading-Strategie zu verwirklichen werden, oft Onlinebroker verwendet, über die resultierende Orders direkt mit vergleichsweise niedrigen Spesen abgewickelt werden können und algorithmisches Trading überhaupt erst prak-

[1] http://www.esignal.com/default.aspx?tc=
[2] Preise s. http://www.esignal.com/esignal/pricing.aspx?tc=

tisch möglich wird. Interactive Brokers (IB)[3] ist ein erwähnenswertes Beispiel eines solchen. Für IB ist bereits ein Test-Account vorhanden, über den Transaktionen virtuell durchgeführt werden können.
Im Bereich des algorithmischen Tradings sind bei high-performance Anwendungen C, C++ und deren Derivate sehr beliebt, bei Anwendungen, die nicht innerhalb von Sekundenbruchteilen operieren, wird die .NET Umgebung unter Windows häufig verwendet.
Dazu stehen im Projektteam mehrere Arbeitsplätze mit .NET-Umgebung zur Verfügung.

2.2 IST-Zustand

Wie bereits erwähnt wurde, ist das algorithmische Handeln eine riesige internationale Industrie. Die meisten Banken haben in der Investment-Abteilung Projekte im Bereich der Entwicklung von Algorithmen zum Wertpapierhandel. Gut funktionierende Algorithmen werden in der Regel aber geheim gehalten, um sich nicht in die Karten blicken zu lassen und den institutionellen Gewinn zu optimieren. Sollte die Handelsstrategie publik werden, wäre es leicht sie auszunutzen und in ihr Schwachstellen zu finden.
Große Institutionen wie Banken sind aber längst nicht die einzigen, die solche Programme nutzen und entwickeln. Mit der zur Verfügung stehenden Technologie kann heutzutage jeder mit Programmierkenntnissen und dem nötigen finanzwirtschaftlichen Wissen Trading-Software entwickeln. Das führt dazu, dass Fonds und selbst private Investoren diese Schiene des Investment nutzen.

Neben der proprietären Software gibt es auch einige wenige Anbieter, die teilweise Handelsentscheidungen im Abo verkaufen. Etwa: Managed-Account Lösungen, denen der Zugriff auf das Investment-Portfolio gestattet wird oder Integrated Development Environments (IDEs), die eine Broker-Schnittstelle zur Verfügung stellen und wo der verwendete Algorithmus selbst programmiert werden kann.

2.2.1 Metatrader

Aktuell bietet Metatrader mit der Trading-Plattform Metatrader5 eine Komplettlösung, die primär auf den Handel auf dem Foreign Exchange Market (FOREX) ausgelegt ist, aber auch Futures, Options und Aktien handeln kann. Die Funktionen sind dabei recht umfangreich, es werden alle üblichen Order-Typen unterstützt. Technische Hilfsmittel wie Indikatoren werden außerdem zur Verfügung gestellt.
Das Interface bietet eine simple Möglichkeit zwischen mehreren Accounts zu wechseln und auf diesen mit unterschiedlichen Handelsstragien zu verfahren. Umge-

[3] http://www.interactivebrokers.com/ibg/main.php

kehrt ist es auch sehr einfach mit einem einzigen Account auf verschiedenen Märkten gleichzeitig zu traden.

Neben manuellem Handel aufgrund von Chart-Analyse und technischen Indikatoren gibt es auch eine Möglichkeit, diesen zu automatisieren und mithilfe von sogenannten *Expert Advisors* Kaufentscheidungen zu generieren. Zusätzlich stellt Metatrader die proprietäre Programmiersprache *MQL5* zur Verfügung, mit der bestehende Bestandteile verändert werden können und neue Expert Advisors programmiert werden können.[4]
In einem Test von Forextraders schneidet die neue Plattform durchgehend gut ab, wobei besonders die ergänzten Indikatoren gelobt werden. [**?**]

2.2.2 GodmodeTrader

Ein Anbieter, der Chart-Analysen und aktuelle spezifische Investmenttipps anbietet ist GodmodeTrader. Es werden sowohl gratis Artikel zur Verfügung gestellt, als auch eine Mitgliedschaft (Gold-Member für €9,90 monatlich) angeboten, die Zugriff auf kostenpflichtige Trading-Services und Expertentipps erlaubt.
Die Gold-Member-Mitgliedschaft beeinhaltet elaborierte Chart-Analysen und Empfehlungen zur Durchführung des Risikomanagements. Mit *Pattern Scout* wird ebenfalls eine automatische Überwachung von verschiedenen Kursformationen und -entwicklungen angeboten.

Im Vergleich zu Metatrader bieten GodmodeTrader theoretisches Wissen und Empfehlungen an; Entscheidungen können in einem Muster-Depot durchgespielt werden. Bei Metatrader liegt der Fokus stärker auf dem praktischen Abwickeln von Transaktionen, auch algorithmisch.

2.2.3 Managed Account

Ohne auf einen speziellen Managed-Account Anbieter genauer einzugehen, soll die Funktionsweise kurz erklärt werden. Der Kunde hat dabei ein Konto bei einer Bank für das er einem Trader eine Handelserlaubnis erteilt. Dieser kann das Geld weder abheben, noch überweisen, sondern nur Trades durchführen. Dabei sollte das zusätzliche Risiko nicht außer Acht gelassen werden.
Die festgelegt Mindesteinlage auf dem Konto variiert zwischen verschiedenen Anbietern, kann aber von einigen zehntausenden bis in die hunderttausende Euro oder mehr reichen.
Die Honorierung des Account-Managers erflogt dabei fast ausschließlich über eine Gewinnbeteiligung und kann in einem höheren zweistelligen Prozentbereich

[4] http://www.metaquotes.net/en/metatrader5

liegen. (vlg. [**?**])

2.3 SOLL-Zustand

Es soll also einerseits eine Software entwickelt werden, mit der man mit einem Handelsalgorithmus (zB.: Moving Average (MA)) an der Börse über einen Broker (zB.: IB) handeln kann. Andererseits soll eine Website erstellt werden, die es ermöglicht, in das Verhalten der Software einzugreifen und zusätzlich den momentanen Status des Handels abzurufen. Diese beiden Teile müssen also mit verschiedenen Schnittstellen miteinander verbunden werden.

Auf der Website soll der verwendete Algorithmus durch Charts dargestellt werden, damit ein Benutzer auch sehen kann, wie die Software handelt und entscheidet. Es soll dort immer mindestens einen Account existieren, der die Software steuern kann, man soll auch mehrere Benutzer hinzufügen können, dies wäre dann aber mit zusätzlichen Kosten verbunden. Außerdem soll man News angezeigt bekommen können, die von einem Newsfeed angeboten werden (zB.: RSS). Es wird möglich sein, dass der User Einstellungen von der Software treffen kann und das Investitionskapital einstellen kann.

Damit die Software erfolgreich und gewinnorientiert funktionieren kann, muss gewährleistet sein das man die momentanen Kursdaten bezieht. Die Kaufentscheidungen sollen mithilfe eines Handelsalgorithmus errechnet werden. Diese Entscheidungen sollen danach ausgegeben und in eine Datenbank gespeichert werden. Natürlich wird dies nur dann ausgeführt, wenn eine Protokollierung erforderlich ist, also bei einer Kauf-/Verkaufsentscheidung. Um auch bei einem Ausfall des Programms die Entscheidungen nicht zu verlieren, wird bei Systemstart eine Logdatei erstellt, in der die Kaufentscheidungen mit Zeit und Datum mitgeführt werden.

Um eine Kommunikation zwischen den beiden Teilprodukten zu gewährleisten, wird eine Website-Software-Schnittstelle implementiert, mit der es möglich sein wird, eine direkte Verbindung zu erstellen. Mit dieser Schnittstelle kann der Benutzer der Website einfach das Handeln der Software unterbinden oder wieder starten, eine Aktie seiner Wahl zum Handeln auswählen und auch die Art des Kaufentscheidungsmodus verändern. Dieser Modus wird zwei Möglichkeiten anbieten, entweder wird automatisch gehandelt und der User wird über dieses Handeln zwar informiert, kann aber nicht widersprechen, oder der User wird bei jeder Kauf-/Verkaufsentscheidung benachrichtigt und trifft dann die Entscheidung selbst. Mit dieser Schnittstelle soll es zusätzlich ermöglicht werden, die Laufzeitparameter der Software im zeitgleichen Betrieb zu verändern und anzuwenden.

Es wird außerdem eine Datenbankverbindung zwischen der Software und der Website existieren, um die berechneten Daten der Software an die Website in regelmäßigen Abständen zu übertragen.

2.3.1 Weiterbildung der Projektteammitglieder

Im Laufe des Projekts wird die Teamfähigkeit und die soziale Kompetenz aller Projektteammitglieder gestärkt. Außerdem wird ihnen das Konzept der Börse und die damit verbundenen Algorithmen näher gebracht. Weiters erhalten die Mitglieder des Projekts neue Erkenntnisse in dem Bereich der Software-/Websiteprogrammierung, vor allem wenn diese parallel arbeiten sollen.

KAPITEL 3

Produktfunktionen

/F010/ **Charts darstellen**

Beschreibung	Charts zur Darstellung der Algorithmen werden auf der Website angezeigt. Dabei muss besonders auf die Schnittstelle zur bereits bestehenden Software geachtet werden. Diese kann zum Beispiel über Webservices oder Datenbanken implementiert werden.
Aufwand	Hoch
Nutzen	Hoch
Ziel	Darstellung von Charts mit den Algorithmusberechnungen auf der Website.
Vorbedingungen	/F040/ Website-Software-Schnittstelle implementieren
Nachbedingungen	-

/F020/ **Account-Management bereitstellen**

Beschreibung	Über zumindest einen primären Account kann sich ein Benutzer anmelden, um sämtliche Controlling-Funktionen der Webseite durchführen zu können.
Aufwand	Mittel
Nutzen	Hoch
Ziel	Benutzer sollen Accounts erstellen können.
Vorbedingungen	-
Nachbedingungen	-

/F030/ **News anzeigen**

Beschreibung	Wenn möglich können Nachrichten-Headlines, die gehandelte Aktien beeinflussen, auf der Webseite angezeigt werden. Ein Beispiel wäre der Bloomberg-Newsfeed, der auch vom Programm benutzt werden könnte.
Aufwand	Mittel
Nutzen	Mittel
Ziel	Anzeigen von einschlägigen News auf der Website.
Vorbedingungen	-
Nachbedingungen	-

/F040/ **Website-Software-Schnittstelle implementieren**

Beschreibung	Um die Software von der Website aus kontrollieren zu können, wird eine Schnittstelle benötigt, diese kann zum Beispiel über Datenbanken, Streams oder Webservices realisiert werden.
Aufwand	Hoch
Nutzen	Hoch
Ziel	Software und Website sollen kommunizieren können.
Vorbedingungen	-
Nachbedingungen	-

/F041/ **Investitionskapital einstellen**

Beschreibung	Die Höhe des Kapitals, das zur Investition freigegeben ist, kann je nach Aktie von der Website aus in der Software eingestellt werden.
Aufwand	Gering
Nutzen	Mittel
Ziel	Einstellung des Investitionskapitals der Software.
Vorbedingungen	/F040/ Website-Software-Schnittstelle implementieren, /F200/ Laufzeitparameter ändern
Nachbedingungen	-

/F042/ **Handeln starten**

Beschreibung	Das Handeln des Programms soll mittels einer Usereingabe auf der Website gestartet werden können.
Aufwand	Gering
Nutzen	Hoch
Ziel	Starten des Handelns der Software von der Website aus.
Vorbedingungen	/F040/ Website-Software-Schnittstelle implementieren, /F200/ Laufzeitparameter ändern
Nachbedingungen	-

/F043/ **Handeln stoppen**

Beschreibung	Das Handeln der Software soll mittels einer Usereingabe auf der Website gestoppt werden können.
Aufwand	Gering
Nutzen	Hoch
Ziel	Stoppen des Handelns von der Website in der Software.
Vorbedingungen	/F040/ Website-Software-Schnittstelle implementieren, /F200/ Laufzeitparameter ändern
Nachbedingungen	-

/F044/ **Entscheidungsmodus ändern**

Beschreibung	Der User muss von der Website aus zwischen einem automatischen und einem manuellen Modus wählen können; bei dem automatischen Modus soll die Software automatisiert entscheiden und handeln; bei dem manuellen Modus wird der User mittels einer Benachrichtigung (siehe /F160/) zum Entscheiden aufgefordert.
Aufwand	Mittel
Nutzen	Hoch
Ziel	Man soll von der Website aus in der Software zwischen einem manuellen und einem automatischen Handelsmodus wählen können.
Vorbedingungen	/F040/ Website-Software-Schnittstelle implementieren, /F200/ Laufzeitparameter ändern
Nachbedingungen	-

/F045/ **Benachrichtigungen verschicken**

Beschreibung	Dem User wird zum Beispiel mittels SMS, E-Mail oder Push-Benachrichtigung der Handelsvorschlag zugesandt.
Aufwand	Mittel
Nutzen	Mittel
Ziel	Der User wird über Handelsvorschläge informiert.
Vorbedingungen	/F040/ Website-Software-Schnittstelle implementieren
Nachbedingungen	-

/F046/ **Aktie auswählen**

Beschreibung	Der User kann auf der Website aus einer prädefinierten Liste die gewünschte Aktie zum Handeln mit der Software selektieren.
Aufwand	Gering
Nutzen	Mittel
Ziel	Möglichkeit von der Website, die Aktien mit denen die Software handelt auszuwählen.
Vorbedingungen	/F40/ Website-Software-Schnittstelle implementieren, /F200/ Laufzeitparameter ändern
Nachbedingungen	-

/F110/ **Input speichern**

Beschreibung	Die vom Datenprovider (als Bars) ausgesendeten Daten sollen gespeichert werden, um später weiter verarbeitet werden zu können.
Aufwand	Gering
Nutzen	Hoch
Ziel	Speichern der historischen und aktuellen Aktiendaten vom Datenprovider.
Vorbedingungen	/F180/ Kursdaten beziehen
Nachbedingungen	/F120/ Bars weiterleiten

/F120/ **Bars weiterleiten**

Beschreibung	Der bisherige Kurs in Form von Bars wird an den Rechenkern übergeben.
Aufwand	Gering
Nutzen	Hoch
Ziel	Transfer der Daten aus „/F110/ Input speichern“ in den Rechenkern.
Vorbedingungen	/F110/ Input speichern
Nachbedingungen	/F130/ Entscheidung berechnen

/F130/ **Entscheidung berechnen**

Beschreibung	Der Rechenkern soll aufgrund des Entscheidungsalgorithmus berechnen, wie das Wertpapier am besten zu behandeln ist (buy, sell, hold).
Aufwand	Hoch
Nutzen	Hoch
Ziel	Kauf- oder Verkaufsentscheidung berechnen.
Vorbedingungen	/F120/ Bars weiterleiten
Nachbedingungen	/F140/ Entscheidung ausgeben

/F140/ **Entscheidung ausgeben**

Beschreibung	Die berechnete Entscheidung wird aus dem Rechenkern in den Verarbeitungskern geleitet, wo sie einerseits ausgegeben und andererseits ausgeführt werden.
Aufwand	Gering
Nutzen	Hoch
Ziel	Anzeige und Durchführung der Entscheidung.
Vorbedingungen	/F130/ Entscheidung berechnen
Nachbedingungen	/F150/ Wertpapier kaufen oder /F160/ Wertpapier verkaufen

/F150/ **Wertpapier kaufen**

Beschreibung	Das System kauft über den Online-Broker-Account eine bestimmte Anzahl an Wertpapieren.
Aufwand	Hoch
Nutzen	Hoch
Ziel	Kaufen eine Wertpapiers.
Vorbedingungen	/F140/ Entscheidung ausgeben
Nachbedingungen	-

/F160/ **Wertpapier verkaufen**

Beschreibung	Das System verkauft über den Online-Broker-Account eine bestimmte Anzahl an Wertpapieren.
Aufwand	Hoch
Nutzen	Hoch
Ziel	Verkauf eines Wertpapiers.
Vorbedingungen	/F140/ Entscheidung ausgeben
Nachbedingungen	-

/F170/ **Log erstellen**

Beschreibung	Das System legt beim Programmstart lokal ein Log-File an, indem alle Entscheidungen versehen mit Datum und Uhrzeit während des Programmlaufs eingetragen werden.
Aufwand	Gering
Nutzen	Mittel
Ziel	Erstellung eines Log-Files.
Vorbedingungen	/F150/ Wertpapier kaufen oder /F160/ Wertpapier verkaufen
Nachbedingungen	-

/F180/ **Kursdaten beziehen**

Beschreibung	Die Software muss eine Schnittstelle zum Beziehen von sowohl aktuellen als auch historischen Bars besitzen. Diese kann zum Beispiel über e-Signal oder Interactive Brokers realisiert werden.
Aufwand	Hoch
Nutzen	Hoch
Ziel	Beziehen der Daten von einem der genannten Datenprovider.
Vorbedingungen	-
Nachbedingungen	/F110/ Input speichern

/F190/ **Datenbankverbindung bereitstellen**

Beschreibung	Das Programm muss eine Datenbankschnittstelle erhalten und die berechneten Daten für die Website erreichbar machen.
Aufwand	Hoch
Nutzen	Hoch
Ziel	Datenbankschnittstelle zur Sicherung von Daten implementieren.
Vorbedingungen	-
Nachbedingungen	-

/F200/ **Laufzeitparameter ändern**

Beschreibung	Um die Controlling-Funktionen zu ermöglichen, muss das Programm flexibler parametrisierbar sein. Es sollen zum Beispiel die Einstellungen auch während der Laufzeit verändert werden können.
Aufwand	Hoch
Nutzen	Hoch
Ziel	Änderung von Software-Parametern zur Laufzeit ermöglichen.
Vorbedingungen	-
Nachbedingungen	-

/F210/ **Multi-Threading ermöglichen**

Beschreibung	Die Software muss Multi-Threading unterstützen, um die Entscheidung über automatisches Kaufen und Verkaufen von verschiedenen Aktien gleichzeitig zu treffen.
Aufwand	Mittel
Nutzen	Hoch
Ziel	Mehrere Aktien sollen gleichzeitig beachtet werden.
Vorbedingungen	-
Nachbedingungen	-

KAPITEL 4

Technische Machbarkeit

4.1 Programmiersprachen

Die Tradingsoftware kann man in jeder erdenklichen Programmiersprache schreiben Allerdings ist es wichtig daran zu denken, dass das Programm effizient arbeitet, da es hardwarenah rechnen soll. Zugleich ist zu berücksichtigen, dass das Projektteam mit manchen Programmiersprachen keinerlei Erfahrung hat.
Die allgemeine Funktionalität muss das Einlesen von Echtzeitdaten der Börse und das Berechnen der Kaufentscheidung beinhalten. 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 reine F#-Lösung.
Bei der Kombination agiert C# als Handlungs- und Steuerkern, als auch zur Kommunikation mit der Website und dem News-Feed und F# als funktionale Programmiersprache, als Rechenkern und „Mastermind“ der Applikation, welche die Entscheidungen trifft. Hierbei wird einerseits eine enorm hohe Arbeitsgeschwindigkeit ermöglicht, da die beiden Sprachen relativ hardwarenah agieren, 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**		**F# R*G**		**C#F# R*G**	
Einfachheit	Aufwand Coding	10%	3	30	2	20	1	10
	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	2	12	1	6
	Gesamt	30%	3	81	2	54	1	27
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	40	2	35
Kosten	Lizenzen	10%	1	10	1	10	1	10
	Support	5%	3	15	1	5	2	10
	Betriebskosten	5%	2	10	3	15	1	5
	Dokumentation	5%	1	5	3	15	2	10
	Gesamt	15%	2	40	3	45	1	35
Dokumentation	Verfügbarkeit	10%	2	20	3	30	1	10
	Vollständigkeit	10%	3	30	2	20	1	10
	Qualität	10%	2	20	3	30	1	10
	Gesamt	30%	3	80	3	80	1	30

Kapitel	Gewichtung	**C++**		**F#**		**C#/F#**	
Einfachheit	30%	3	81	2	54	1	27
Leistung	15%	1	15	3	40	2	35
Kosten	15%	2	40	2	45	1	35
Dokumentation	30%	3	80	3	80	1	30

Gesamtbewertung			
Endreihung	2	3	1

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

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

4.2 Websprachen

Bei der Wahl der Technologie zur Umsetzung der Web-Controlling-Oberfläche muss auf einige Kriterien geachtet werden, um zu einer Entscheidung zu kommen. Die Sprache muss es ermöglichen, eventuell durch API-Zugriff, hinreichend komplexe Charts zu generieren, um alle gewünschten Indikatoren und Preisentwicklungen nahe der gewünschten Form darzustellen. Außerdem muss ein Zugriff auf eine mit der Software gemeinen Datenbank möglich sein, um Kurswerte und eventuell andere berechneten Daten zur Darstellung abzufragen.
Zur Durchführung der Webfunktionalität kommen folgende Sprachen in die engere Auswahl.

- PHP
- ASP.NET
- Java Servlet mit JSP

Der augenscheinliche Vorteil von PHP ist die bereits gesammelte Erfahrung des Projektteams mit dieser Sprache. Zusätzlich sind die Chartingfunktionen beispielsweise durch die pChart Bibliothek bereitgestellt. [1] pChart bietet verschiedene Varianten zur Darstellung an. Unter anderem Line-Charts, Candle-Charts und Indikatoren.

ASP.NET bietet den Vorteil einer komplett homogenen Integration der Website mit der Software, da sowohl ASP.NET, als auch C# die Windows .NET-Library benutzt. Zur Kommunikation kann Windows Communication Foundation (WCF), eine Sammlung von Kommunikationswerkzeugen zwischen verteilten Systemen innerhalb von .NET. Außerdem ist, angenommen man bleibt in der Windows-Welt, sowohl die Implementierung, als auch die Separation of Concerns (SoC) einfacher umzusetzen. Mit Hilfe von WCF kann der Aufruf von entfernten Methoden sogar über JavaScript geregelt werden, um eine noch stärkere logische Trennung zwischen Software- und Websitefunktionalität zu schaffen.
Die Chartingfunktionen können seit .NET-Version 3.5 direkt mit dem inkludierten Framework „Chart Controls for .NET“ und dem ASP-Chart-Tag umgesetzt

[1] urlhttp://www.pchart.net/

werden. Das Framework unterstützt unter anderem auch „Advanced Financial Charts“.[2]

Die Variante mit Java Servlets und JSP ergab sich in der Nachforschung als ungünstig, da zur homogenen Umsetzung die Software mit Java implementiert werden müsste, was u.a. aus Performancegründen ausgeschlossen wurde. Ohne die Logikklassen in Java zu programmieren, ergibt sich aus diesem Ansatz kein Vorteil.

Das wahrscheinlich wichtigste Kriterium zur Wahl der Sprache ist die Kommunikation zwischen der C# Software und der Website. Für eine genauere Beschreibung der Datenschnittstellen siehe 4.3.

4.3 Datenschnittstellen

Grundsätzlich soll es zwei große Datenschnittstellen geben. Die eine befindet sich zwischen der Software und der Website und ist hauptsächlich für die Bereitstellung der Controlling-Funktionen notwendig. Die andere besteht eigentlich wieder aus zwei einzelnen Schnittstellen, zur Kommunikation jeweils der Website und der Software mit der Datenbank.

4.3.1 Schnittstellen zwischen Software und Website

Diese ist die kompliziertere Schnittstelle, da an sie hohe Anforderungen gestellt werden. Sie soll hoch leistungsfähig und gleichzeitig einfach zu implementieren sein. Für diese Art von Schnittstellen kommen grundsätzlich einige Varianten in Frage. Es wird nun versucht, jede dieser Varianten inklusive ihrer Vor- und Nachteile zu erläutern und anschließend eine Conclusio zu treffen, welche die Sprache ergeben soll, die sich am Besten eignet.

Die erste Möglichkeit wäre CGI. Damit können von der Website aus, lokale Zugriffe, über das Ausführen von Commandline-Befehlen, abgesetzt werden. Dies wäre grundsätzlich kein schlechter Ansatz, das größte Problem hierbei ist leider nur, dass man die Website und die Software auf dem gleichen physikalischen Server laufen lassen muss. Außerdem kann es dabei zu Problemen mit den Zugriffsrechten kommen. Zusätzlich entstehen Unsicherheiten, da ein Hacker beispielsweise nur die Website knacken muss und damit auf den gesamten Server zugreifen kann.

Die zweite Realisierungsmöglichkeit wäre über Streams, die an Sockets angehängt werden. Dabei müssen sich allerdings ebenfalls der Webserver und der Server mit

[2] urlhttp://www.microsoft.com/en-us/download/details.aspx?id=11001#overview

der Software im selben Netzwerk befinden. Dies kann zwar über neue Mechanismen umgangen werden, wird aber noch nicht als schöner Stil angesehen und ist damit für unseren Verwendungszweck eher ungeeignet.

Eine dritte Möglichkeit wäre es, diese Schnittstelle ebenfalls über eine Datenbank zu realisieren. Dabei müsste die Website die Controlling-Parameter auf der Website abspeichern und die Software fragt z.B. einmal in der Sekunde nach, ob schon neue Controlling-Parameter vorliegen. Dies wäre an sich keine so schlechte Möglichkeit, da für die Übermittlung der Kursdaten ja bereits sowieso eine Datenbank vorliegt und man diese einfach erweitern könnte. Allerdings entspricht die Performance dieser Lösung nicht ganz unseren Vorstellungen, da die Software wie gesagt immer nachfragen müsste, ob Änderungen vorliegen. Dadurch wird einerseits die Dauer verlängert, bis die Controlling-Operation durchgeführt wird und andererseits entsteht ein nicht zu vernachlässigender Aufwand durch die regelmäßigen Anfragen an die Datenbank.

Eine Realisierung dieser Schnittstelle wäre auch über einfache Webservices möglich. Dies würde, genauso wie die Realisierung über eine Datenbank, eine Kommunikation über das Internet ermöglichen. Allerdings sind einfache Webservices aufwändig zu implementieren. Für die Windows-Umgebung, auf die das Projektteam abzielt, gibt es allerdings eine einfach Möglichkeit zur Implementierung von solchen Websites.

Diese heißt WCF. Es handelt sich dabei um ein sehr mächtiges Framework von Microsoft, dass z.B. Webservices angenehm implementieren lässt. Es hat eine Unzahl an Funktionen, ermöglicht eine Kommunikation über das Internet und ist zudem extrem leistungsfähig und vergleichsweise simpel in der Implementierung. Außerdem ist es von Microsoft und kann somit perfekt sowohl in unsere C#/F#-Umgebung als auch in unsere ASP.NET-Webumgebung integriert werden. Aufgrund all dieser Vorteile ergibt sich also die Realisierung der direkten Schnittstelle zwischen Website und Software mittels WCF als die favorisierte Möglichkeit.

4.3.2 Schnittstelle der Software und der Website zur Datenbank

Dass diese Schnittstelle in einer einfachen Form realisiert werden kann, ist eigentlich klar, da es auch wenig sinnvolle Alternativen dafür gibt. Worin man allerdings unterscheiden kann, ist im gewählten Database Management System (DBMS). Es liegt hierbei eigentlich nur die Wahl zwischen MySQL und PostgreSQL vor, da das Projektteam nur in diesen beiden Erfahrung hat und es auch keine anderen preislich und leistungsmäßig ansprechenden Alternativen gibt.

Zwischen MySQL und PostgreSQL entscheiden eigentlich hauptsächlich die Lizenzen. MySQL kostet bei kommerzieller Nutzung Geld. Wohingegen PostgreSQL Open-source ist und auch bei kommerzieller Nutzung kostenfrei bleibt. Aufgrund des stark beschränkten Kapitals entscheidet das Projektteam also hier klar für PostgreSQL. Außerdem besitzt das Projektteam, wie bereits erwähnt, schon Erfahrung im Umgang mit PostgreSQL und somit wird ein Erlernen eines neuen SQL-Dialekts auch nicht benötigt.

KAPITEL 5

Wirtschaftliche Machbarkeit

5.1 Aufwandsabschätzung

5.1.1 Personalaufwand

Aus der Abbildung 5.1 ist ersichtlich, dass die FP-Analyse ca. 120 Punkte ergibt. Anhand der Tabelle aus Abbildung 5.2 ergibt sich für 120 Function-Points eine ungefähre Projektdauer von 4 Personenmonaten. Dies entspricht bei einer Umrechnung von 40 Stunden pro Woche eine Gesamtstundenanzahl von 640 Arbeitsstunden. Dies wiederum geteilt durch die Anzahl der Projektteammitglieder ergibt das eine Workload von ca. 213,3 Stunden pro Projektteammigtlied.

Bei einem Stundensatz von 50,00 € pro Arbeitsstunde ergibt dies ein benötigtes Kapital von 32.000,00 € für das gesamte Projekt Aquila.

5.1.2 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 folgende:

- 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 Lizenz, zur Modellierung von Diagrammen.

Kategorie	Anzahl	Klassifizierung	Gewichtung	Summe
Eingabedaten	0	einfach	2	0
	1	mittel	3	3
	2	komplex	5	10
Abfragen	3	einfach	2	6
	4	mittel	3	12
	3	komplex	5	15
Ausgabedaten	0	einfach	3	0
	1	mittel	4	4
	3	komplex	6	18
Datenbestände	0	einfach	6	0
	2	mittel	9	18
	2	komplex	14	28
Referenzdaten	0	einfach	4	0
	0	mittel	6	0
	0	komplex	9	0
Summe			E1	114

Einflussfaktoren (können den errechneten Function Point-Wert um +/- 30% ändern)			
	1	Verflechtungen mit anderen Anwendungssystemen (0.5)	4
	2	Dezentrale Daten, dezentrale Verarbeitung (0.5)	4
	3	Transaktionsrate (0-5)	5
	4	Verarbeitungslogik	
	a	Rechenoperationen (0-10)	4
	b	Kontrollverfahren (0-5)	3
	c	Ausnahmeregelungen (0-10)	3
	d	Logik (0-5)	2
	5	Wiederverwendbarkeit (0-5)	3
	6	Datenbestands-Konvertierungen (0.5)	4
	7	Anpassbarkeit (0-5)	3
Summe der 7 Einflussfaktoren		E2	35
Faktor Einflussbewertung E3 = (E2 / 100) + 0,7	E3	1,05	
Bewertete Funtion Points E1 x E3			119,7

Abbildung 5.1: FP-Analyse zu Aquila

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 5.2: IBM Umrechnung von Function-Points in Personenmonate aufgrund von Erfahrungswerten

- 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 €.

5.1.3 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.

Die Nutzung von Aquila soll auch auf einem Server nahe der NASDAQ getestet werden. Dafür muss ein Server gemietet werden. Billige Pakete beginnen hier bei ca. 20,00 € für Server mit unseren Mindestanforderungen. Dies beläuft sich für die Projektlaufzeit von einem Jahr auf ca. 240,00 €.

Dadurch beläuft sich das gesamte benötigte Investitionskapital auf ca. 6.920,00 € für die gesamte Projektdauer.

5.2 Risikoanalyse

5.2.1 Personenausfall

Eintrittswahrscheinlichkeit: gering
Auswirkungen: gering

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 Auftraggebers plötzlich kein Interesse an der Umsetzung des Produktes mehr gegeben ist, und es dadurch zu einem extremen Zeitverzug kommt, der bis zum Abbruch führen kann.

Folgende präventive Maßnahmen werden eingeführt:

- Regeln für den Umgang innerhalb des Projekts
- Ausreichendes Interesse jedes Mitglieds, und keine leistungstechnischen Probleme
- Gutes Verhältnis zu den Auftraggebern

5.2.2 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.

5.2.3 Technische Risiken

Datenverlust
Eintrittswahrscheinlichkeit: gering
Auswirkungen: mittel

Aufgrund der nicht auszuschließenden Gefahr des Datenverlusts, muss dafür gesorgt werden, die Sicherheit der Daten sowie auch ihre Verfügbarkeit zu garantieren. Dieses Problem wird mit Hilfe 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.

5.3 Kosten / Nutzenpotential

Das Ziel des Projektes Aquila ist es einerseits eine Variante zur Automatisierung des Aktienhandels zu schaffen und dabei die volle Kontrolle und Übersicht zu behalten. Andererseits liegt ein langfristiger Nutzen in der Entstehung von Know-how und der technischen Fortbildung des Projektteams; schließlich ist das Kapital Bildung bei der Abwägung des Nutzens nicht außer Acht zu lassen.
Die Tradingsoftware samt Website zur Steuerung soll als einfach bedienbares Komplettpaket angeboten werden. Die Nutzungs- und Vertriebsrechte bleiben bei den Entwicklern, verkauft werden können Lizenzen mit beliebig beschränkter Nutzerzahl zu unterschiedlichen Preisen.
Ebenfalls in Rechnung gestellt wird die Installation der Software auf einem entweder bereits verfügbaren System oder es wird die benötigte Serverarchitektur gemietet. Das Service der Serverbereitstellung ist von Auftragnehmerseite nicht vorgesehen. Sollte dies allenfalls gewünscht werden, wird das Hosting voraussichtlich outgesourced.
Die Tradingsoftware trifft vollständig automatisiert Entscheidungen zum Kauf und Verkauf von Wertpapieren und ermöglicht es somit auch potentiellen Marktteilnehmern mit wenig Zeit systematisch zu handeln, was ohne Software mit erheblichem Zeitaufwand einhergeht. Die Software bleibt dabei ebenfalls stets veränder- und erweiterbar; den Kunden könnten zukünftige, neue Softwareversionen gegen einen Abopreis verrechnet, beim Initialkaufpreis einberechnet oder einzeln verkauft werden. Die Website ermöglicht das gesamte Controlling der Software und bietet neben Konfiguration der Handelseigenschaften Überblick über aktuelle Vorgänge.
Die Software ist auf das Handeln über einen Broker-Account ausgelegt, wodurch eine Zielgruppe von privaten oder institutionellen Einzeltradern oder kleinen Tradinggruppen angesprochen werden. Eben diese Zielgruppe profitiert besonders von der Simplizität der Verwaltung, der Übersicht und dem Erweiterungs- und Verbesserungspotential.

Neben der Lizensierung soll die Software im Unternehmen selbst auch zum Handeln eingesetzt werde. Je nach Kapitalaufwand kann damit der Umsatz gesteigert werden. Durch die eigenständige Nutzung kann zusätzlich noch ein Feedbackprozess installiert werden. Die firmeninternen Trader geben Feedback an die Entwickler, um zukünftige Softwareversionen zu verbessern, was sowohl dem Ruf und folglich Absatz, als auch dem Profit durch das Trading mit der Software zugute kommt.

Zusammenfassend entsteht aus diesem Projekt ein Nutzen in mindestens 3 Bereichen.

- Lizensierung als Gesamtpaket

- Eigenständiges Trading mit der Software
- Fortbildung des Projektteams

Die Preise könnten beispielsweise wie folgt festgelegt werden:

Leistung	**Preis**
Einzelnutzerlizenz	1.999
5-Benutzer Lizenz	4.999
Unbeschränkte Lizenz	9.999
Installationspauschale	299
Installationsarbeiten	100/h

Tabelle 5.1: Preise in €

Bei Gesamtkosten von 44.381 € wären diese bei den empfohlenen Preisen relativ schnell gedeckt.

Durch die Lizensierung der Software bieten sich ebenfalls potentielle Folgeprojekte an, die neue Versionen mit erweitertem Funktionsumfang in Ausschau stellen. Dabei kann sowohl internes, als auch externes Feedback eingebracht werden. Ebenso kann der Algorithmus aufgrund der modularen Integration auf Kundenwunsch verändert werden, oder sogar ein vollständig neuer integriert werden. Aufgrund dieses großen Potentials, muss durch dieses Projekt selbst kein signifikanter Gewinn entstehen.

KAPITEL 6

Projektorganisation

Der Grund dafür, dass dieses Projekt ins Leben gerufen wurde, sind die Projektauftraggeber Professor Mag. Hans Brabenetz und Professor Dr. Helmut Vana. Der Projektleiter ist Gabriel Pawlowsky und das Team besteht noch zusätzlich aus zwei weiteren Entwicklern: Peer Nagy und Josef Sochovsky.

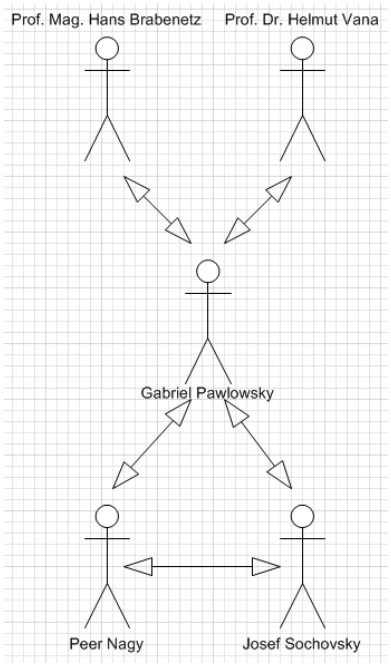

Abbildung 6.1: Projektorganisationsdiagramm erstellt mit Microsoft Visio

Die Zuständigkeiten der Projektteammitglieder sind folgendermaßen verteilt:

- Peer Nagy
  Entwicklung der Website
- Gabriel Pawlowsky
  Führendes Projektmanagement
  Entwicklung der Software
- Josef Sochovsky
  Implementation der Schnittstellen

Das Testing des Endproduktes wird von dem Team als Gruppe bearbeitet.

KAPITEL 7

Projektplanung

7.1 Projektstrukturplan

	PSP-Code	Vorgangsname	Dauer	Anfang	Fertig stellen	Vorgänger
1	**1**	**Aquila**	**106 Tage**	**Mi 14.11.12**	**Mi 10.04.13**	
2	**1.1**	**Projektmanagement**	**106 Tage**	**Mi 14.11.12**	**Mi 10.04.13**	
3	1.1.1	Projektstart durchführen	1 Tag	Mi 14.11.12	Mi 14.11.12	
4	1.1.2	Projektkoordination	99 Tage	Do 15.11.12	Di 02.04.13	3
5	1.1.3	Projektcontrolling durchführen	99 Tage	Do 15.11.12	Di 02.04.13	3
6	1.1.4	Projektmarketing durchführen	99 Tage	Do 15.11.12	Di 02.04.13	3
7	1.1.5	Projektabschluss durchführen	6 Tage	Mi 03.04.13	Mi 10.04.13	6
8	**1.2**	**Software**	**92 Tage**	Do 15.11.12	Fr 22.03.13	
9	1.2.1	Struktur implementieren	20 Tage	Do 15.11.12	Mi 12.12.12	
10	1.2.2	Datenanbindung implementieren	20 Tage	Do 13.12.12	Mi 09.01.13	9
11	1.2.3	Broker implementieren	14 Tage	Do 10.01.13	Di 29.01.13	10
12	1.2.4	Controller implementieren	21 Tage	Mi 30.01.13	Mi 27.02.13	11
13	**1.3**	**Website**	**92 Tage**	Do 15.11.12	Fr 22.03.13	
14	1.3.1	Informationsdesign durchführen	10 Tage	Do 15.11.12	Mi 28.11.12	3
15	1.3.2	Graphikdesign durchführen	7 Tage	Do 29.11.12	Fr 07.12.12	14
16	1.3.3	Datenbank einrichten	6 Tage	Do 29.11.12	Do 06.12.12	14
17	1.3.4	Website implementieren	37 Tage	Di 08.01.13	Mi 27.02.13	20
18	**1.4**	**Schnittstellen**	**70 Tage**	Do 13.12.12	Mi 20.03.13	**9**
19	1.4.1	Datenbank-Software-Schnittstelle implemenieren	11 Tage	Do 06.12.12	Do 20.12.12	16
20	1.4.2	Datenbank-Website-Schnittstelle implementieren	22 Tage	Fr 07.12.12	Mo 07.01.13	16
21	1.4.3	Website-Software-Schnittstelle implementieren	11 Tage	Do 28.02.13	Do 14.03.13	12
22	**1.5**	**Testing & Abschluss**	**18 Tage**	**Fr 15.03.13**	**Mi 10.04.13**	
23	1.5.1	User-Acceptance-Test durchführen	5 Tage	Fr 15.03.13	Do 21.03.13	21
24	1.5.2	Securitytesting durchführen	5 Tage	Fr 15.03.13	Do 21.03.13	21
25	1.5.3	Unit-Testing durchführen	7 Tage	Fr 22.03.13	So 31.03.13	24
26	1.5.4	User-Testing durchführen	7 Tage	Fr 22.03.13	So 31.03.13	24
27	1.5.5	Fehlerbehebung durchführen	7 Tage	Mo 01.04.13	Di 09.04.13	26
28	1.5.6	Abnahme durchführen	0 Tage	Mi 10.04.13	Mi 10.04.13	27

Abbildung 7.1: Projektstrukturplan

7.2 Balkenplan

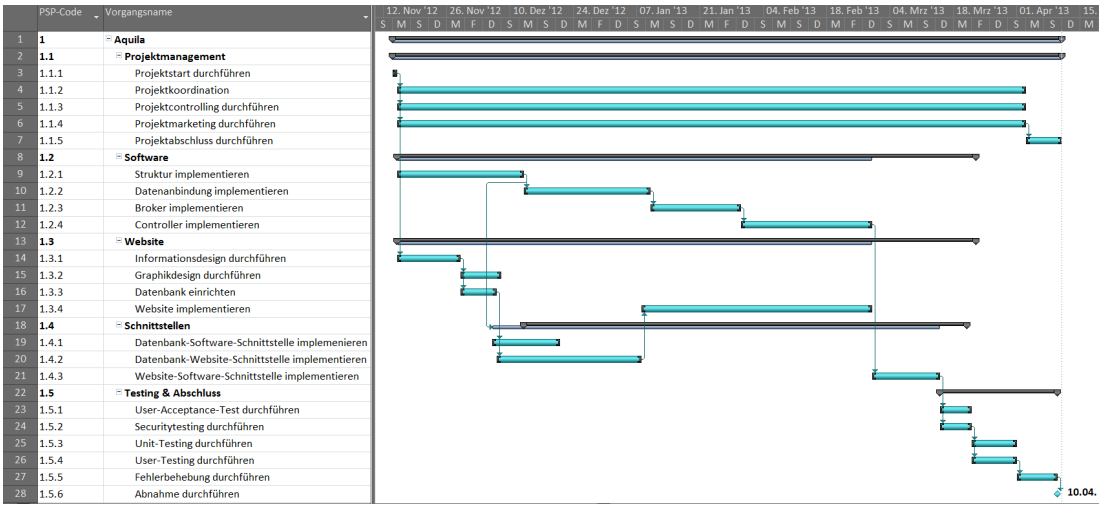

Abbildung 7.2: Balkenplan

7.3 Meilensteinplan

Meilenstein	**Deliverable**	**Datum**
Projektstart		14.11.2012
Datenbank eingerichtet	Script zum Erstellen der Datenbank + dazugehörige Dokumentation	06.12.2012
Externe Schnittstellen implementiert	Schnittstellen zur Kommunikation mit eSignal und InteractiveBrokers + dazugehörige Dokumentation	29.01.2013
Software & Website fertiggestellt	Tradingsoftware mit simplem Algorithmus und Websiteoberfläche und Charting + dazugehörige Dokumentation	27.02.2013
Produkt fertiggestellt	Version des Produkts (inklusive aller internen Schnittstellen), die noch nicht getestet wurde + dazugehörige Dokumentation	14.03.2013
Testing abgeschlossen	Testberichte und etwaige Verbesserungen am Produkt	09.04.2013
Projektabnahme	Ausgefülltes Abnahmeprotokoll	10.04.2013

KAPITEL 8

Management Summary

Nach ausführlicher Beschäftigung mit den kritischen Themen sind für das Projekt AQUILA geeignete Lösungswege entwickelt worden und es wurde aus mehreren potentiellen Varianten die günstigste ausgewählt. Folglich ist das Projekt durchführbar.

Aus der Variantenbildung ergab sich die Wahl der Programmiersprachenkombination C#/F# für die Software, da dadurch die Performance der Programmierung und der Ausführung gleichermaßen gegeben ist. Die Webschnittstelle soll in ASP.NET implementiert werden, da dadurch die Funktionen der .NET-API weitergehend verwendet werden können. Weil beide Technologien aus einer Hand kommen, kann WCF für die Schnittstelle zwischen beiden Komponenten verwendet werden.

Die Gesamtkosten des Projektes wurden auf 44381 € veranschlagt. Empfohlene Preise für die Software liegen bei 1999 € für eine Einzelbenutzerlizenz, 4999 € für eine 5-Benuter Lizenz und 9999 € für eine unbeschränkte Lizenz.

Der geschätzte Aufwand des Projektes beträgt 213.3 Stunden, der auf 3 Projektteammitglieder aufgeteilt wird. Der geplante Projektzeitraum ist vom 14.11.2012 bis 10.04.2013.

Glossar

AFA Abschreibung für Abnutzung. 25

DBMS Database Management System. 21

DDE Dynamic Data Exchange. 3

FOREX Foreign Exchange Market. 4

FP-Analyse Function-Points-Analyse. v, 23, 24

GIT GIT-Server. 25, 26

IB Interactive Brokers. 4, 6

IDE Integrated Development Environment. 4

MA Moving Average. 6

WCF Windows Communication Foundation. 19, 21, 33