5 Implementation

5.1 Overview

While many mathematical models concentrate on the prediction of stock prices, few use these predictions to manage a portfolio given the theoretical basis described in Chapter III. I have developed a system, NELION, based on non-linear stock prediction models and this investment theory. It uses the many different possibilities that the Internet offers both for data retrieval as well as user interaction.

NELION is designed to manage the portfolios for numerous investors and to suggest customized purchases and sales for each, in an effort to achieve an optimal portfolio. Investors can request to receive portfolio recommendations on a daily, weekly or monthly basis, depending on their preference. Correspondingly, these recommendations are based on mathematical models, which take into account daily, weekly or monthly stock data.

As described in Chapter 4, NELION is divided into four components that are shown in Figure 4.1.1. The database is a Microsoft SQL Server 7 running on MS Windows 2000 and the Task Agent and Administration Tool are two applications written in Microsoft Visual C++ 6.0. The User Interface runs

on MS Internet Information Server and uses Active Server Pages. All components connect to the database via an ODBC interface, which is used to manage the entire data pool. Both Visual C++ programs require the ODBC Data Source Name (DSN) of the NELION database as an input parameter and use MS Windows authentification to ensure access rights.

5.2 The HTML Interface

The HTML Interface runs on MS Internet Information Server 5.0 and provides the investor with a means to manage his portfolio. Since this web server is not connected to the Internet continuously, the domain www.nelion.net is hosted on an Internet service provider and when the web server logs onto the Internet it updates a link on the welcome page to it. After a successful log in using the e-mail address as a user name and a password, the investor is shown an overview his current portfolio.

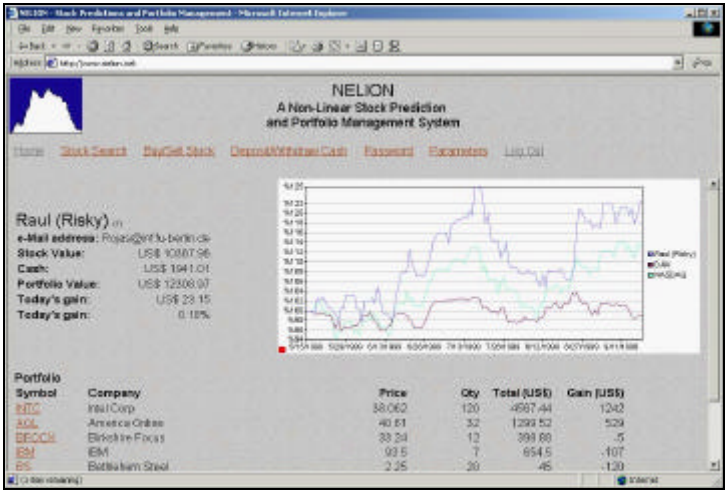

Figure 5.2.1: Porffolio Overview via the HTML Interface

The upper half of the page is devided into two columns: The left side contains basic portfolio information including the e-mail address, the sum of all stocks currently owned, the cash reserves in the portfolio and the sum of these two figures, which represents the total value of the portfolio. It also shows the gain or loss in portfolio value both in US dollar and in percentage terms. The right side contains a graph comparing the return of the portfolio with the developments of the Dow Jones Industrial Index and the Nasdaq.

Below these two components, the web page shows a list of all stocks in the current portfolio, including the ticker symbol, the full stock name, the current price and quantity as well as the product of these two values, which represents the total

investment in this stock. Lastly, the table shows the total gain or loss that the investor has incurred with this stock.

Finally, the web page contains a further table with the current recommendation for the investor. This component contains the same column as the portfolio table, with the exception of the gain. A positive quantity represents a buy, a negative quantity a sell recommendation.

From this page, the investor can select numerous links that permit him to manage and adjust his portfolio.

5.2.1 Stock Search

The stock search link opens a new window that permits the entry of a stock ticker symbol, a company name or a part of a company name. After completing this information and pressing the “Search” Button, NELION will attempt to locate a stock with the specified stock symbol.

If it can not find the corresponding company, it will list all companies that contain the word specified in the search field. The ticker symbol for each company in the list also serves as a link to the relevant stock, so that the investor can view all relevant company details.

5.2.2 Buy/Sell Stock

This link is used to update the portfolio after a transaction. NELION will open a dialog in the menu window when the link is pressed and will request the investor to specify whether a stock was bought or sold. Additionally, it needs the transaction date, the ticker symbol, the amount, purchase or sales price as well as the transaction costs. The latter will be defaulted with the value specified in the investor parameters.

5.2.3 Deposit/Withdraw Cash

If the investor changes the cash reserves by depositing or withdrawing cash from the account, he will need to update the portfolio accordingly with this link. The system will respond with a dialog in the menu window requesting information on whether cash was deposited or withdrawn from the account, the transaction date and the cash amount.

5.2.4 Password

In order to change the NELION password, an investor has to enter the current password and type the new password in twice. The double entry is necessary since none of the passwords are shown in clear text but only with a star (“*”) for every character entered.

5.2.5 Parameters

This page is used to specify all investor preferences and parameters. For each parameter, the system offers an explaination on the usage of this parameter. Besides the investor name, e-mail and SMS e-mail address, the system requires information on the investment horizon and the e-mail frequency. The transaction cost field is used to populate the "Buy/Sell Stock" dialog with a correct default value. The remaining parameters focus on the risk adversity of the investor and include the minimum transaction amount, as well as the volatility, error, volume and correlation risk adversity.

This same page is used for the online application of new investors. It allows a new applicant to specify all relevant parameters when he opens an account without intervention by the NELION administrator.

5.2.6 Log Out

The log out link removes a session token from the web server so that no additional transactions are possible and returns the investor to the NELION welcome page.

5.3 The Administration Tool

The NELION Administration Tool is the primary tool for the administrator of the system. It contains all common features

of an MS Windows application including a menu, icon bar for short cuts and a status bar at the bottom of the screen.

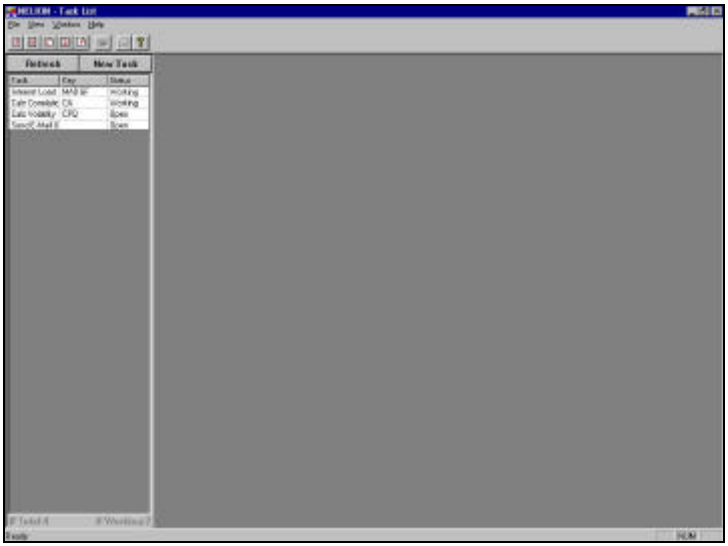

Figure 5.3.1: The NELION Administration Tool

The program is written using a multi document interface (MDI) so that the administrator can open multiple windows, each displaying information for a specific investor or stock. Additionally, it is possible to open a window for the system parameters. The task list is also implemented as a MDI window but, unlike the others, it remains open on the left side of the screen as long as the application is running.

Following the MS Windows standard, the Administration Tool uses multiple tabs in documents in order to make optimal use

of the available screen real estate. Some of these allow the user to enter or update data pertaining to the corresponding entity, while others show historical, calculated data or parameters. They are described in detail in the following sections. Full screen prints are included in Appendix C.

5.3.1 Investors

The “General” tab for the investors contain the basic investor information like the name, e-mail address, investment horizon, e-mail notification interval, transaction costs and minimum transaction amount, as well as all investor risk adversity parameters. The investor type is maintained here as well and for test investors the data entry fields for the beginning and end of the test have to be entered. The investor number cannot be modified and represents the internal system number. Similarly, the portfolio value is calculated from the current stock prices held by the investor.

The “Portfolio History” tab displays a graph of all the stocks that the investor owned since he started tracking his account on NELION. The proportional value of each stock as well as the overall value of the portfolio is visible at a glance for the entire portfolio history.

On the “Purchases” tab, the Administration Tool provides a list of a purchases and sales undertaken within the portfolio. Each transaction is one record and includes the ticker and

company name, date, stock price, amount, and the product of these two, representing the total transaction value.

The “Portfolio” tab displays a list where every record represents one stock that is currently held. Each line contains the ticker and stock name, the current stock price, the amount held and again the product of these two, representing the total value of the stock. The sum of these values is shown on the header of this column representing the total portfolio value. Finally, the list contains a column for the value gained or lost with this stock.

The “Return” tab shows the actual return on the investment for three different periods: Since the portfolio was included in NELION, since the beginning of the calendar year and in the last 12 months.

5.3.2 Stocks

The “General” tab for a stock includes the ticker and stock name and the web site for data download. At present, the latter only offers one option, which is “Quote Central.” If more sites provide historic stock data, especially for European and Asian stocks, this list and the additional required functionality would be expanded. The ticker name is used to create a table on the database to store the daily price and volume information for the stock.

The two additional pieces of information on the tab, current price and volatility are read-only fields since the former is downloaded from the Internet and the latter is calculated.

For a new stock in NELION, this is the only tab available. When editing an existing stock, three further tabs with information are available. The “Models” tab shows a list with the header information of the mathematical models stored for this stock. Each model has a name, the number of data input values used, the prediction interval as well as the NMSE.

The “Graph” tab displays the price movement over time of the stock. The graph auto scales the y-axis to ensure that the entire data is visible on the screen.

Finally, the “Correlation” tab shows a list for the correlations of the chosen stock with all other stocks in the system. The list is sorted in decreasing order, so that the selected stock will always be at the top with a correlation of “1”.

5.3.3 Parameters

The “Parameters” data entry screen permits the maintenance of the system parameters. The “Refresh Interval” parameter controls how often the Administration Tool will update the task list. The entry is interpreted in seconds, so that a value of 60 will result in a refresh rate of once a minute.

The “Bank Interest Rate” represents the return on cash kept with the broker or with another bank. Since this can be viewed

as a risk-free investment, any stock purchase, which is by definition risky to a greater or lesser degree, is only recommended by the system if the prediction for it exceeds the bank interest rate.

The “Time Difference to New York” parameter records the time difference from the current location to the home of the New York Stock Exchange. Since “Quote Central”, the Internet source of our data, does not update the historic stock price and volume list until approximately 4:00 a.m. local time in New York, we use this value to calculate the time for the Internet data download.

The “SMS Threshold” defines a band of uncritical price swings. If the price of a stock changes by an amount, which exceeds the SMS threshold, all investors who own the stock as well as the NELION system administrator are notified through an e-mail. The e-mail address is different from the one used for regular updates and should be tied to a messaging provider that forwards the e-mail to a specified mobile phone via short messaging system, SMS. The system administrator’s SMS e-mail address is defined in the corresponding system parameter.

5.4 The Database

The NELION database server runs on MS Windows 2000 and employs MS SQL Server 7.0. This software infrastructure ensures scalability to a multi-gigabyte installation, while permitting an individual investor to work with standard PC

hardware on a single, modern workstation. The database platform includes functionality to ensure referential integrity, by including counters, stored procedures, triggers, primary and foreign keys and other constraints.

In order to harness the power of these tools effectively, I used S-Designor DataArchitect 5.1 to design and document the database, which only required a few manual adjustments for installation. The graphical tool permits relating tables graphically and generates an SQL script for different database types. The database creation is then limited to the execution of this script and assigning user access rights.

DataArchitect also allows the definition of data types, which map to types supported by the underlying database. In a first design, I defined a type "Geld"[2], which mapped to the data type "money". At a later stage, I reduced the number of different data types and mapped "Geld" to "float", in an effort to reduce complexity. This merely required creating a new database and transferring the data to it, with an appropriate mapping from "money" to "float".

MS SQL Server offers scheduled tasks, which I used to automatically generate tasks in the task list for e-mail updates of the portfolio value and recommendations for the investors. Similarly, the system inserted an "Internet Download" task into the task list once a weekday. In order to adjust the

[2] "Geld" is the German word for „money."

mathematical models to the changing dynamics of the market, NELION autonomously increased the error of all models by 5% every Sunday. This gave new models calculated by the genetic algorithm a better chance of undercutting the existing models, with no loss of generality. In a worst-case scenario, it identified a model with the same parameters as optimal again, so that it merely overwrote itself.

A trigger on the “Purchases” table secured referential integrity between transactions, investor portfolio and the investor header tables. When a purchase or sale was entered, either through the Administration Tool or directly on the database, the trigger updated the “Portfolio” table and reduced the portfolio value on the investor table to reflect the associated transaction cost.

Since the data tables for each stock were created dynamically and represented the bulk of the tables in the database, I did not include the same trigger on each of these to keep the current value of the header table updated. Instead, I wrote a stored procedure that was called from the Administration Tool and updated both the data table and the stock header table when data was downloaded from the Internet.

A number of tables used to store the mathematical models contained a database-internal counter as the primary key, since this key was a foreign key in subsequent tables. This design reduced the data requirements in the database and normalized the tables effectively. When storing a model on the database, it was necessary to save the model header

information and then to retrieve the counter because it was a required piece of information on the detailed tables. Here, I also used stored procedures to save the record and immediately deliver the ID as the return value, because it reduced the database accesses from two to one, thereby optimizing the system.

Though the insert function automatically increments the counter for a table, I manually added a stock with the counter “0”, which is defined as “Cash”. It is used to maintain the cash held by the individual investors in their respective portfolios and is automatically adjusted to account for purchases and sales. Similarly, on the Purchase table, each transaction results in an entry for this “stock” to account for the transaction costs, as they are defined for each investor.

5.5 The Task Agent

The Task Agent is a separate program with a simple interface that consists of an “Exit” button as well as nine check boxes, which allow the user to define which actions this instance of the program should perform. When the program exits, the current configuration is stored in the MS Windows registry and is restored the next time the program is started to ensure that the same configuration is retained.

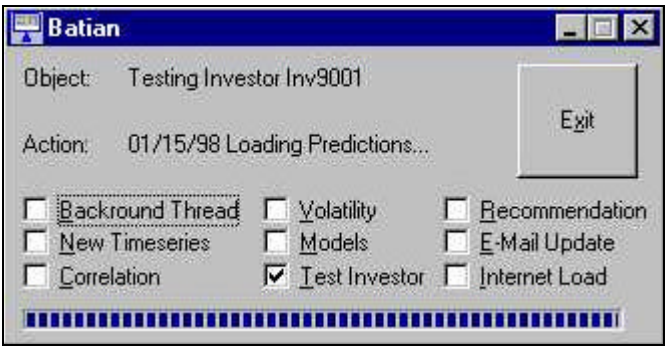

Figure 5.5.1: The Task Agent Program

Two text boxes provide the user with a description of the current object and the actions on that object that are currently executed. The objects are either the stock or an investor from the corresponding table.

The server process typically “sleeps” until either the Administration Tool or a scheduled task enters a record in the task list. In this state, it could start a second thread that benefited from the idle processing time to search for improved prediction models using a genetic algorithm. While performing this optimization, a progress bar is updated every second to show that the system is still running and checking for new tasks every six seconds.

If it found an “Open” task in the task list, it marked the task as “Working” so that no other Task Agent on the network begins executing them. After performing the relevant task, it deleted the task from the task list and the cycle starts afresh.

The Task Agent can perform nine different functions, which are described in detail in the following section.

5.5.1 Internet Load

The Internet Load function assumes that a permanent connection to the Internet exists and downloads data from a site called “Quote Central”, which has its Internet homepage at www.wallstreetcity.com. Navigating to the “Stocks” tab on this site allows the user to enter a stock ticker name to retrieve stock data and charts with a 20-minute delay. On the left-hand side of the screen, it also provides the option to view historical quotes. Choosing this function, the site provides options to enter the stock, the start and end dates of the historic data requested, the data interval as well as eight different formats for the historical stock data. After selecting the appropriate parameters, pressing the “Get Quotes” button performs the request.

By analyzing the URL that results from this request, one can automate the retrieval by generating it within NELION and executing it directly. The Internet Load function builds the correct URL, using a daily interval and requests historic stock data in the “HEADER 5 – Super Charts” format. In order to limit the data download, it sets the start date to the day of the last date available in the system and overwrites it in case it has been updated. All new data is appended to the stock data

table. For newly created stocks, NELION attempts to download all data since January 1, 1980.

After sending the request to the Internet site, wallstreetcity.com responds with a new page, which contains a temporary link to a page with the “Historical Quote Header 5 Export File”. NELION extracts this link and executes it. The result is a comma deliminated ASCII file, which contains the date, open, high, low and close prices as well as the daily volume. From this page, it extracts the close price and volume for each day and stores them on the database. For days, which do not have an entry on the web site, like public holidays and weekends, NELION generates a fictitious record using the previous stock price and volume figures. In case the price changes more than a specified percentage from one day to the next, the system automatically sends an alert to all investors, who own the stock as well as the system administrator. The e-mail address of this alert can be different from the address used for regular updates, so that the investor receives it as an SMS notification on his mobile phone. The technical details of this function are described in section 5.5.5 below.

Since the system does not download any images, the data volume of the download is kept at a minimum so that the daily update per page can be accomplished within one second, given the necessary bandwidth to and within the Internet. Since the system is tailored to this specific site map, it is clear

that a change in the structure at wallstreetcity.com will require an adjustment of the NELION Internet Load function

Though other sites also offer historic stock quotes, many are fee-based and require a log in process, which complicates the retrieval. Of those that were free of charge none that I found offer the simple ASCII format and several tests showed that parsing an HTML page is considerably more complex and error prone.

It is worth noting that the system does not retrieve dividend and stock split notices, so that these would have to be entered manually. The former would merely be an adjustment to the available cash. For the latter, stocks are normally split in a ratio of 2:1 or 3:1, so that one can expect a drop in the stock price by approximately 50% or 66% respectively. The investor database administrator will be alerted to drastic changes in price through the SMS notification so that he can take appropriate action.

5.5.2 Calculate Volatility

Calculating the volatility of a stock is a straightforward implementation of the equation described in 4.5.2. To this end, NELION retrieves all of the data of a particular stock and executes the loop described in the section. This task is performed when a new stock is created and can be repeated periodically thereafter.

A single task in the task list, for stock ID “0” performs this for all stocks. Since this stock counter refers to “Cash” and no data table exists for this dummy stock, the instruction is well defined.

5.5.3 Calculate Models

Before a forecast for a stock can be calculated, it is necessary to calculate predictors. The Calculate Models task performs this task for a specific stock for all investment horizons (one day, one week and one month) and all model types (ARN, ANN, KNN and MM), keeping only the two models for each combination that have the lowest NMSE. The algorithm for the model calculation follows from section 4.5.3.

It is worth noting that though the database only stored all model and financial parameters as floats, all models employed the data type double in RAM, in order to ensure greater precision for internal calculations, potentially increasing the quality of the results.

Given the models as a basis, the system was able to make predictions for each horizon. In an effort to continually improve the quality of the models, however, the fine-tuning of these predictors took place in the genetic algorithm implemented in the background thread described in section 5.5.9.

5.5.4 Calculate Correlations

To calculate the correlations for a stock, the system loaded the historic data into memory, excluding weekends. It then iteratively loaded the data for the remaining time series into memory and started calculating the correlation as specified in section 4.5.4 starting with the later of the two beginning dates. The function did not disregard public holidays, because this does not significantly affect the calculation and helps reduce the complexity of the application.

5.5.5 Send E-Mail Update

The Send E-Mail Update function is designed to keep the investor informed on the status of his portfolio. The subject of the message contains the current total account value and in parentheses the change from the last e-mail.

The body of the e-mail contains a table with all the stocks that the investor currently holds in his portfolio. For each stock, the message shows the number of shares, the current price and the product of these two, representing the value held in this stock. The last column shows the total gain or loss that the investor has incurred to date with this investment. The last row is always the current cash holdings of the portfolio. At the bottom of the table, the message shows the sum of all investments representing the total portfolio value.

All dollar amounts are displayed with three significant digits, to ensure that eights of dollars (US$ 0.125) can be represented accurately.

NELION generates a plain-text e-mail and needs a MAPI compliant e-mail client to send it. The application uses the default profile to access the e-mail services. The MAPI client does not have to be running, but its behavior and settings determines how the message is treated thereafter. I tested the task agent with Outlook Express 5 running under Windows 2000 as well as Outlook 2000 running under Windows 2000 and Windows 95. The OS did not affect the operation, but although both e-mail clients were set to send e-mails immediately, only Outlook Express 5.0 performed this action, even if it was not started. Outlook 2000 needed to be started, but only sent the message because it was configured to send and receive new messages every ten minutes.

In order to provide a consistent appearance, I registered the domain name NELION.NET. All e-mail was sent using the e-mail address Administrator@NELION.NET.

5.5.6 Calculate Recommendation

In the recommendation calculation for a specific investor, NELION attempted to load the prediction with the specified investment horizon for each stock into memory. If the prediction did not exist on the database, it proceeded to load all historic stock data and the model with the lowest NMSE

into memory and calculated the prediction. To make this prediction available for subsequent recommendations, it was stored on the database.

With the predictions in memory, NELION identifies the stocks, which promise to achieve an annual return bigger than the bank interest rate and uses the algorithm described in section 4.5.5 to identify the optimum portfolio.

The investor subsequently receives an e-mail similar to the e-mail update described in the previous section. In addition, however, the system adds a line for each purchase or sale recommendation, specifying the stock and the recommended number of shares. For Auto-Investors, the transactions were simulated, using the most recent stock price and included transaction costs in order to provide a realistic scenario for comparison. In this case, the e-mail indicated exactly which transactions were performed.

5.5.7 New Time Series

This task combines the tasks that are necessary for each new time series: Internet Load, Calculate Volatility, Calculate Models and Calculate Correlations. This was necessary in case more than one Task Agent accessed the database, because combining these tasks into a single function ensured that they were executed sequentially. If they were executed in parallel, one Task Agent might be downloading the historic data from the Internet, while a second might start calculating

its volatility, models or correlations with incomplete data, which would have falsified the results.

5.5.8 Test Investor

The Test Investor function is necessary to identify parameter combinations, which result in investment recommendations that suit the preferences of each investor. The implementation follows from the theoretical discussion in section 4.5.8.

In order to provide a realistic test environment, I created a separate database, deleted all stock data after the beginning of the test interval, and recreated all prediction models. Thereafter, task agents on two computers spent two weeks optimizing the models using the genetic algorithms of the background thread.

The test investor results are discussed in detail in chapter 6, where the experimental results show the quality of our Internet trading system.

5.5.9 Parameter Selection with the Genetic Algorithm

If the Background Thread check box was set on the interface and the task list of the database only contained functions that the instance of the Task Agent was not requested to perform, it started a program thread to calculate new predictors for a randomly selected stock.

In the foreground, the application continued updating the progress bar every second and checked the task list every six seconds. In case a new task was entered while the background thread was active or the Exit button was pressed, the program completed the calculation of the current model in the background before ending the thread.

The symmetric multi-processor (SMP) architecture of Windows 2000 ensured that the two threads were executed on different processors in multi-processor machines so that no degradation in execution speed was noticeable. On a single processor computer, the two threads had to share the resources, so that two computing intensive tasks increased the execution time.

The additional models calculated by the background task used the genetic algorithm described in section 4.5.9, using inheritance and cross-over to generate new model parameters and then checking whether this model was able to achieve a lower NMSE than either of its parents. If this was the case, it replaced the worse of the two existing models and became a parent for future calculations. This regenerative process was facilitated by increasing the NMSE value of all models by 5% every Sunday, because this permitted new models, which capture new market dynamics, to replace outdated predictors.