

Charles University in Prague
Faculty of Mathematics and Physics

BACHELOR THESIS



Tomáš Helešic

Investment Strategies Simulator

Department of Software Engineering

Thesis supervisor: RNDr. Ondřej Šerý

Study program: Computer Science, General Computer Science

2010

I would like to thank my supervisor, RNDr. Ondrej Šerý, for his time and valuable guidance during my work on the thesis. Also, I would like to thank my friends for their comments, which forced me to look into my work from different perspectives, and a big thanks goes to my parents for their endless support.

I declare that I have written this bachelor thesis on my own and entirely using the cited sources. I agree to lending of this thesis and its publishing.

In Prague, December 8th, 2010

Tomáš Helešic

Contents

1	Introduction	6
1.1	Trading on the stock market	6
1.2	Project goals	7
2	Analysis of automatization strategies	9
2.1	The market and stock exchange	9
2.2	The financial market valuation	11
2.3	Technical analysis	12
2.3.1	Charts	12
2.3.2	Technical indicators	13
2.4	Investment strategies	13
3	Implementation and the application model	15
3.1	Data acquisition and storing	16
3.2	Charts	17
3.3	Technical indicators	17
3.4	Strategy representation and evaluation	18
3.4.1	Compilation and communication with the application	18
3.4.2	Request for data	19
3.4.3	Evaluating the strategy	19
3.4.4	Strategy result	20
3.5	Error processing	21
4	User Documentation	22
4.1	Installation	22
4.2	Starting the program	23
4.3	Change application settings	24
4.3.1	Change a database	24
4.3.2	Create a new database	24

4.3.3	Select data provider	25
4.3.4	Set up initial date for downloading	25
4.4	Add or delete assets	25
4.5	Chart	25
4.5.1	Display chart	26
4.5.2	Operate with chart	26
4.5.3	Zooming and scrolling	27
4.6	Apply indicator	27
4.7	Apply simple strategy	28
4.8	Create and run strategy scripts	28
4.8.1	Create a strategy	28
4.8.2	Create and run strategy scripts	29
5	Conclusion	31
5.1	Comparison with another available implementation	31
	Bibliography	33
	A Scripts' template	36
	B CD contents	40

List of Figures

1.1	Technical analysis in use (taken from [4])	7
2.1	Example of the supply and demand curve (Taken from [7]) .	10
2.2	Example of upward and downward trend	10
2.3	Two examples of chart representation	13
3.1	Internal model	15
4.1	The layout of the main program	23
4.2	Application settings	24
4.3	Add and delete assets	26
4.4	Set up a simple strategy	27
4.5	Set up a simple strategy	28
4.6	Select a script	29
4.7	Example of a strategy result	30

Název práce: Simulátor obchodních strategií
Autor: Tomáš Helešic
Katedra (ústav): Katedra softwarového inženýrství
Vedoucí bakalářské práce: RNDr. Ondřej Šerý
e-mail vedoucího: ondrej.sery@d3s.mff.cuni.cz

Abstrakt: Předmětem této práce je vytvoření simulátoru obchodních strategií. Výsledný program umožňuje uživatelům stahovat aktuální i historická burzovní data, vizualizovat je pomocí grafů a implementovat na ně prostředky technické analýzy. Tyto komponenty jsou navrženy a propojeny tak, aby vytvořily plnohodnotné prostředí pro psaní, vyhodnocení a zobrazení uživatelských strategií.

Klíčová slova: Obchodní strategie; Technická analýza; Skriptování;

Title: Investment Strategies Simulator
Author: Tomáš Helešic
Department: Department of Software Engineering
Supervisor: RNDr. Ondřej Šerý
Supervisor's e-mail address: ondrej.sery@d3s.mff.cuni.cz

Abstract: The goal of this thesis is the creation of an investment strategies simulator. The resulting program allows users to download current and historical stock data, visualize it using charts and implement on it the tools of technical analysis. These components are designed and linked to create fully worthwhile environment for the creation, evaluation and representation of user strategies.

Keywords: Investment strategies; Technical analysis, Scripting;

Chapter 1

Introduction

1.1 Trading on the stock market

Trading on the stock exchange is an appealing way of investing funds. The trading can be done in commodities, foreign, stock and future exchange. The stock market brings profit by speculating on the rise or fall of the market, but speculating is not the only way of gaining profit - holding shares can also bring money from dividends. Dividends are the cash consideration of joint stock companies, paid to its shareholders, which means that the portion of company profits are paid out to stockholders [1]. *Profit* and only *profit* is what the trader is looking for. It implies searching for better strategies to get ahead of other traders. Due to the fact that the market is evolving rapidly, the secure strategy for gaining profit does not exist. Success is achieved by understanding the market and practicing, but nothing is guaranteed. The success in trading is a combination of wisdom and common sense. *When we trade, we have to try to achieve both, realizing that wealth can be a result of wisdom* [2]. Technical analysis (Fig. 1.1) provides only a recommendation, based on a survey of the market. It is a general advice made by skilled brokers, but this advice still does not assure profit. The final decision has to be made by the trader and here again it comes down to practice. Even a good strategy applied in the wrong time can lead to bankruptcy.

Another very important fact is that the majority of people who trade are not successful enough to justify being in the market. If the broker trades with the majority, then he will be as good as the average, and the average does not make money. [3]. This forces one not to be too dependent on analysis of others.

Do something different, even if it's wrong.

Charles B. Goodman

The next section describes the key components of the project, which can improve the trader's skills and make him more successful in real-time trading.



Figure 1.1: Technical analysis in use (taken from [4])

1.2 Project goals

The main goal of this thesis was to create an application which enables the testing of different strategies on the historical stock exchange data. The application contains methods and structures allowing the user to:

- download and store stock exchange data from various sources
- create and apply technical indicators
- create and evaluate strategies
- chart these records, technical indicators and results of applied strategies

The next chapter analysis the automatization process of investment strategies. Chapter 3 describes the analysis and decisions which had to be made during the project development. This process covered dividing the assignment into smaller modules, their processing and creation of appropriate interfaces. The greatest challenge was to design and implement the modules' interfaces such that they cooperate in a clear and simple way. This chapter also explains the steps needed to extend the program. Chapter 4 explains the installation process, running the program and recommendations on how to achieve maximum serviceability. The last chapter concludes and summarizes the effort in analyzing and processing the given task, providing a comparison with the available programs of similar purpose, and discusses the possibilities of expanding the program in the future. Appendix A contains a template of the script and describes the process of its creation. Appendix B contains a list of contents on the accompanying CD.

Chapter 2

Analysis of automatization strategies

This chapter contains the definition of market behavior, positions, fundamental and technical analysis, which determine the trading strategies and their automatization.

2.1 The market and stock exchange

The market is a space where the exchange of economic goods and cash is made. The developed economies do not have the barter type of trade, where goods are exchanged for other goods directly. The Stock exchange is an institution which organizes the market with investment tools. It is one of the elements of capital market. Exchange takes the form of double auction, where the supply and demand create the final price of the traded instrument[5]. As the supply and demand change, the price moves towards the equilibrium (Fig. 2.1). The buyers are competing and pushing the price up. The buyer compares the price with his marginal utility - if the price is higher than the marginal utility, the buyer loses interest. On the other hand, the sellers, who are also competing amongst themselves, are pushing the price down [6]. This results in upward (bull) and downward (bear) market trends (Fig. 2.2). Bull and bear trends can make the greatest profit, since if a trader goes with the trend, he can continuously make money.

Position in the stock exchange trading is a commitment to buy or sell a financial instrument for a given price. By entering a position, the price and volume of the transaction need to be specified. The volume in financial

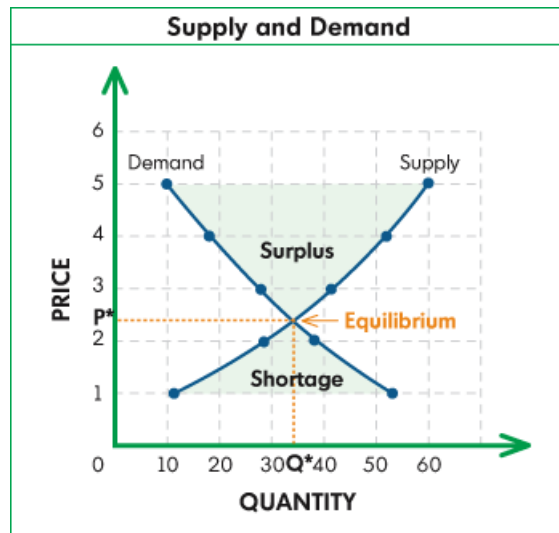


Figure 2.1: Example of the supply and demand curve (Taken from [7])

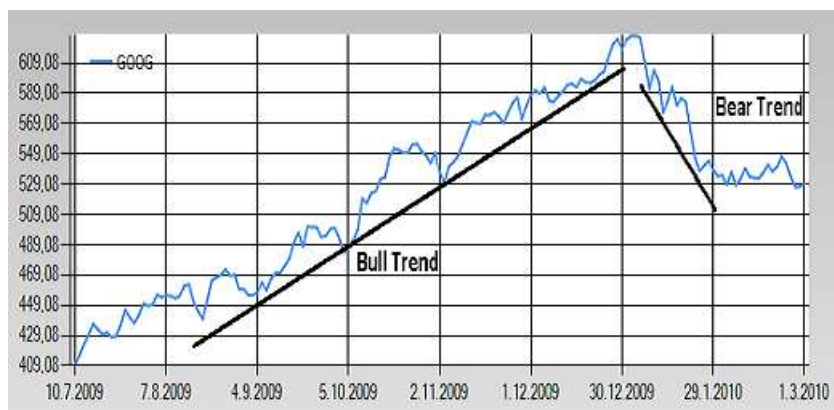


Figure 2.2: Example of upward and downward trend

trading is called a lot - it represents the standardized amount of a traded financial instrument. There are two types of positions:

Long Position

- If the market is in an upward trend, the trader will make a profit
- Buying an instrument opens a position
- Selling closes the position

Short Position

- If the market is in a downward trend, the trader will make a profit
- Selling an instrument opens a position
- Buying closes the position

It is possible to speculate on a downward trend of a financial instrument. The broker can lend this instrument to the trader who obliges to buy the instrument back at a later date in return.

2.2 The financial market valuation

A financial instrument valuation is the process of calculating theoretical values, which in other words means predicting potential market prices. Rate the instrument as undervalued or overvalued so that the market will move in particular trend towards the equilibrium. There are 3 main approaches of market valuation[8]:

- Fundamental analysis
- Psychological analysis
- Technical analysis

Fundamental analysis represents the examination of price-fixing factors and information which are available for the traders. It uses economical, statistical, accounting, historical, political and demographic circumstances as input values. Prices derived from these conditions are compared with the current price and the trader can decide if the particular instrument is worthy of investment [9]. Psychological analysis examines the human behavior during trading. Investing is a collective matter and this analysis concentrates on the psychology of crowds and impulses to mass selling and buying [10].

2.3 Technical analysis

Unlike fundamental analysis, technical analysis uses only data generated by the market, such as price, volume, volatility, number of opened positions and cross-market correlation. It is used to estimate the price trend and the probability of its reversion by searching for price patterns in charts. The technical analysis is based on certain expectations [11]:

- All events and information affecting the market are already covered in the price. This results in a fair price and the analysis can build on it.
- The prices are not moving randomly, but in trends.
- The history tends to repeat. This is caused by traders, whose reactions are similarly often repeated.
- *What* is more important than *Why*. Fundamental analysis tries to explain and predict events, but technical analysis cares only about the price.

Technical analysis critics claim that it lacks sufficient theoretical base in order to be credible. However, some techniques have their own justification. The best practice is to use fundamental analysis to pick an instrument and technical analysis to decide on timing [12].

2.3.1 Charts

The charts are a very useful tool in searching for price patterns such as *Head and Shoulders*[13] and *Double top, bottom reversal patterns* [14].

Line charts display a single value on the Y-axis. This value is, in general, the closing price of an instrument.

Candlestick charts present four values of an instrument in a given time frame. These values are *open*, *close*, *high* and *low*. The filling color of the body represents the relationship between the opening and closing prices, as shown in Fig. 2.3(a).

Bar charts are similar to Candlestick, as they also display *open*, *close*, *high* and *low* prices. The only difference is in their graphical representation (see Fig. 2.3(b)).

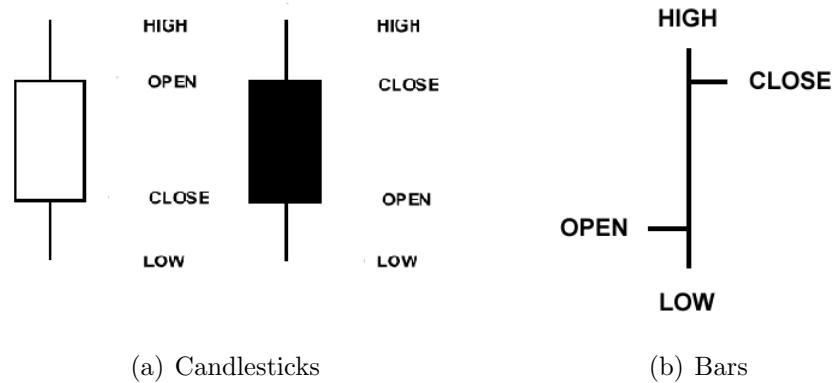


Figure 2.3: Two examples of chart representation

2.3.2 Technical indicators

The technical indicators can be any class of metrics whose value is derived from an instrument's price activity. Indicators strive to estimate the future price and direction by considering past patterns. They filter the small market fluctuations which could be confusing. They are divided into four groups, based on their orientation:

- Volume
- Oscillators
- Trend indicators
- Bill Williams

Technical indicators are widely used in technical analysis, as they simplify the process of evaluating and specifying investment strategies for the trader [15]. Automatization of investment strategies can not be done without technical analysis.

2.4 Investment strategies

Investment strategy is a set of rules, procedures and behaviors, designed to aid in choosing individual investments and creating an investor's portfolio.

Different investment strategies reflect individual investors' personalities and decisions in trying to achieve profit. Some traders may prefer riskier investments to achieve higher returns in a short term (*Momentum investing [16]*), others may choose to invest in an instrument with a consistent long term growth (*Growth [17] and value investing [18]*). The procedure of choosing a strategy is dependent on the investor's requirements. The trading process is simple, but difficult to apply successfully:

1. Create a strategy
2. Find instruments that are consistent with the strategy
3. Create a portfolio

In automating this process, fundamental and psychological analysis can not be used. Only technical analysis and its tools lend themselves well to being transformed into computer algorithms. The strategy can then collect the signals generated by the individual indicators and decide on investing into the given instrument.

Chapter 3

Implementation and the application model

This chapter elaborates on the implementation of the program. It presents the application model, interfaces and external libraries used. Figure 3.1 represents the internal model of the application. The project can be separated into smaller tasks, which are implemented independently. These modules are linked through interfaces to enable future extensibility of the application.

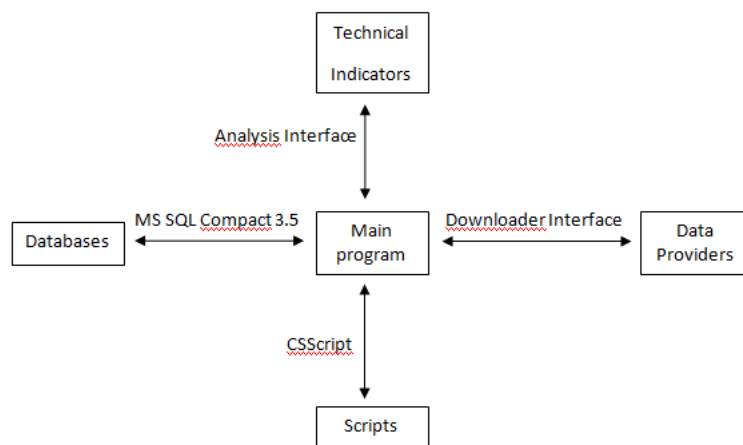


Figure 3.1: Internal model

3.1 Data acquisition and storing

The first task was to gain access to historical stock exchange data. Immediate data generated by the stock exchange are very valuable and highly charged. This program was designed to test strategies in long or relatively short terms. The immediate data were therefore not necessary. Yahoo and Google finance servers provide daily data. Both enable the user to download historical data in a *.csv* format. All the historical values of stocks are available on Yahoo automatically, but non-stock instruments together with all other data from Google are available only for the previous year. The application stores the data in the database. It comes at an additional disk space cost, but this approach is much faster than repetitively downloading data. The user is not expected to have any advanced computer skills, such as managing a database server, and for that reason the underlying database is embedded in the application itself (Microsoft SQL Compact Edition 3.5 [19]). Another benefit of this approach is that the program will work even if there is no internet connection (provided that the data was downloaded to the local database in the past). The database schema consists of only two tables. The first, *CompanyTicks* table, stores names of instruments and their symbols (unique identifiers). The second table contains the instruments records:

- *Date*, date and time of the record, type *DateTime*
- *OpenPrice*, opening price, type *Double*
- *High*, highest price, type *Double*
- *Low*, lowest price, type *Double*
- *ClosePrice*, closing price, type *Double*
- *Volume*, transactions volume, type *Long*
- *AdjPrice*, adjacent price, type *Double*

Adjacent price is sometimes ignored by the data providers. The data can be requested and downloaded via libraries with implemented *DownloaderInterface*. These libraries are dynamically plugged in after launching the application and are stored in a public collection during the program session. Classes *DownloaderServices* and *DownloaderPlugins* are used. The process of dynamically loading the library is the same as with indicators:

1. create an assembly
2. check if the assembly implements the required interface
3. create an instance of the object and add its reference to the collection

DB class implements the methods for managing the database, such as creating, deleting and updating an instrument. It calls a plug-in method, *GetHistory()*, declared by the interface, and receives a table with an instrument records. The program freezes during the process, therefore background worker is created and the process of downloading and updating the database runs in the background. Both plug-ins use the *Webclient* class to download the data. They parse the values and store them in a table.

3.2 Charts

Chart is the base element of technical analysis. Technical indicators are calculated from the instrument values and use either the same or a different chart area to display their records. They also need to be aligned with the instrument chart to be correctly interpreted by the user. These were the main requirements when selecting the appropriate charting control. The Microsoft Chart Control fulfills these demands and provides even more methods for processing and visualizing records, such as setting labels, tool tips, grouping values, zooming and scrolling charts [20]. The instruments and indicators records are stored in a structure called Data table, and these records are binded to the chart data points. When the user chooses a new time frame, the records are refiltered and binded again.

3.3 Technical indicators

Technical indicators implement the technical analysis as is written in the Analysis section. The indicators take the form of a plug-in, which means that they are dynamically loaded when the application starts. They implement the *Analysis Interface*. This enables the user to create new indicators and enrich the possibilities of applying technical analysis in the program. The interface specifies the method called by the application. *Graph*, *GraphType* and *ColumnNames* store the information for MS Chart Control to correctly interpret the indicator in a chart. *Calculate* receives the instruments records

and calculates indicators values. The indicator itself stores calculated values for the current instrument, in order to improve the performance of the application. The program takes the indicator from a collection of linked indicators, calls the *Calculate* method and records together with *Graph*, *GraphType* and *ColumnNames* values as parameters, and calls the *Insert-IntoGraph()* method. This method processes the parameters, aligns values with the instrument chart and displays them into the same chart area or a different one. The application implements 4 indicators: *Simple and Weighted moving average*[21] are Trends indicators, *On Balance Volume*[22] is an example of a Volume indicator and *Relative Strength Index*[23] represents the Oscillator indicator.

3.4 Strategy representation and evaluation

One of the challenges was to create the aforementioned modules in such a way that they provide a solid base for creating and evaluating strategies. The strategy automatization process needs to collect signals from the indicators and instrument records. The task was to analyze and think of a way of compiling the script and aligning it to the interface during runtime, creating a method of passing data to it, evaluating that data and saving the results for further representation. User friendly specification of the script was an important constraint for technically less inclined users.

3.4.1 Compilation and communication with the application

The application compiles and links the scripts during runtime. An alternative solution was to pre-compile the scripts with *csc.exe* and load them during the application launch. This solution needed additional intervention from the user, and was refused, since the user cannot edit and run the script again with different settings. The script needed to be aligned to the interface to clearly specify what can be demanded and what can be offered. The compilation process creates an assembly and this assembly needs to be added into the application domain. If the assembly is loaded into the domain, it cannot be removed. This implies the need to create a temporary domain and unload it after the script is evaluated to enable running different scripts. The temporary domain limits the communication between the script and the pro-

gram such that all method calls can only be made between the host and the script, but the host can use every method from the main application. The CS-Script library enables the main program to compile the script, create a temporary domain, align the script to an interface and unload the domain after the execution [24]. *CSScript.Compile()* method compiles the selected script and creates the assembly into the temporary domain by creating an instance of *AsmeHelper* class. Method *CreateAndAlignToInterface <Istrategy>()* aligns the script to the *Istrategy* interface and creates an instance of the script class. The host can then call methods and set properties in the compiled script. The script can also call methods and set properties to the host, but the host object has to set its reference to the script first *script.host = this;*

3.4.2 Request for data

The script has to inform the application which data it needs for evaluation. All instrument and indicators records are of type *DataTable*. The script therefore needs to specify which records it needs and it does so by specifying the column names. The host method *Demands(params string[] demand)* enables the script to request values for evaluation. This method is implemented with variable parameters so the user is not limited by the number of different indicators. The script sets everything it needs, such as Opened-Price, High, Low, Volume etc.. from an instrument and indicators like RSI, OBV, SimpleMA. The script also has to specify which record will be used for opening the position. *host.PriceColumn = OpenPrice* or other instruments record will do it. The host processes the parameter array by calling the *PrepareRecords()* method. This method removes from an instrument those columns which are not set in the list. The indicators values are merged with the filtered instrument table. These records are filtered by a specific time frame set by the user and are ready for the *EvaluateStrategy()* method.

3.4.3 Evaluating the strategy

There are various methods for evaluating the strategy. The first method is to send all of the records to the script at once, but this method is unsuitable since the script has to implement while or for loops and operate with the *DataRows* and index them. Another disadvantage is that the script has the full image of values and can cheat by using future records. This strat-

egy therefore cannot be applied, as it violates the prerequisites of technical analysis. A better approach is to send only one row at a time without the date time record. The for cycle is then implemented in the host, and the user does not know if the record is one month or 20 years old. The strategy then follows the rules of technical analysis, and can only react to the current or historical records. The data are transferred from *DataRow* into *Dictionary <string, double>* so the user can index them by string, and the script is more transparent. Finally, the user can create the logic in the *Evaluate* method. The logic is based on the instrument and indicators records, it can open or close the position by calling *host.OpenPosition(direction)* or *host.ClosePosition()*. The opening or closing price is picked from the specified record as mentioned in *Request for data* section. For secure trading, the methods *Stop loss* [25] and *Trailing stop loss* [26] are also implemented with the *TakeProfit* [27] method. After being set, this method controls the price of the instrument, and can close the position if the conditions are met. Only a single long or short position can be opened at once. The *StrategyInterface* declares all the methods and properties for script and host as a way of enforcing communication conventions.

3.4.4 Strategy result

The user needs to obtain a clear and transparent result of applying the strategies. Except for the requested records from the script and the opening/closing price, he needs to know which condition has been met, that caused the position to be closed. This is achieved by adding a message into the result table. The method *Stop loss* and several others set the unique identifier and bool value *StopedByFunction* to true when their condition is met. The host can then recognize what caused the close event and output this information. The result table is not sorted and the opening and closing events are not grouped together. The *CalculateProfit()* method groups the related opening and closing events and calculates the total profit, together with the number of Long/Short positions which gained profit or created a loss, and the overall profit and loss for the Long/Short positions. Every transaction costs money, therefore all closed positions deduct a fee, which the scripts sets before execution of its logic. All these records are visualized in the *FrmStrategyOutput* form. Another important task was storing the results of strategies for further comparison with other results. The Excel application is the most suitable for this because of its table representation. The

data are exported into the Excel spreadsheet by filling the cells with result table records and with values calculated in the *CalculateProfit()* method. The conditional cell formatting is used for better visualization. The application uses the *Microsoft.Office.Interop.Excel.dll* library to accomplish this task. Exporting to *.csv* format is used for long term record storing and further evaluation and comparison with other results in future application extensions.

3.5 Error processing

The application distinguishes two types of errors: user errors and runtime errors. User errors can be recovered from, the user is informed about the cause of error, and is given the chance to correct it. Runtime errors are not recoverable and cause the application to exit. Before exiting, the user is also informed about the cause of the crash.

Chapter 4

User Documentation

This chapter presents the end user documentation, which contains installation instructions, as well as instructions for running and operating the program.

4.1 Installation

Target platform for this project is Windows. The installation folder *Setup* contains two files: *Setup.exe* and *StockEval.msi*. *Setup.exe* is the main installation file, which checks the prerequisites, *.Net Framework 3.5 with SP1 and Windows Installer 3.1*, and if they are missing, it tries to connect to the Internet, download these programs and install them. The installation process has to be run with the administrator's privileges. After this first step, the user chooses the target folder for the main program and the installation process is then completed, creating a shortcut to the program on the desktop. *Src* folder contains the source files of the application also with libraries, certificate keys for the assemblies which are necessary to compile the code. The program was written in Visual studio 2008. The main directory of the installed program contains the following folders:

- *\Databases* database files, new databases are created here
- *\DataProviders* libraries used to download stock market data from the remote servers
- *\Indicators* contains technical indicators
- *\Resources* icons

- \Output primary folder for saving records of evaluated strategies in Excel and CSV files
- \Scripts primary folder for loading scripts into the program

4.2 Starting the program

The program is started by running the file *StockEval.exe* which can be found in the main installation directory. After starting the program, it searches for the data providers and the indicators, and loads the default database. If the database is not found, the program opens the Settings form and prompts the user to select the database manually or to create a new one. Changing the settings is described later in this chapter. The main program is shown in figure 4.1. The main form layout:



Figure 4.1: The layout of the main program

1. displays all assets stored in the database
2. enumerates all available technical indicators
3. provides controls which operate the charts

4. displays the chart area
5. provides controls for adding/deleting assets, selecting and applying strategies and changing application settings

4.3 Change application settings

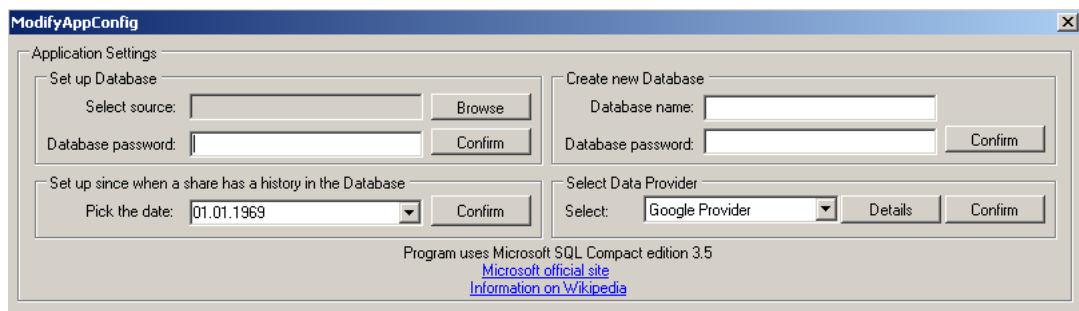


Figure 4.2: Application settings

This form contains some important settings for obtaining and storing data: changing and creating a new database as well as selecting a data provider, which sets the stage for strategy testing.

4.3.1 Change a database

In the Group box named *Set up Database*, clicking on the *Browse* button opens up a form which displays a list of databases from the folder *\Databases*. The user can then select one of these and confirm his selection by pressing the *Select* button. Next, he types the password for the selected database and confirms the change of the database by pressing the *Confirm* button. A message box then appears, containing the confirmation or an error message describing what went wrong.

4.3.2 Create a new database

In the Group box *Create a new database*, the user inputs the name and password for the new database. The password is optional and can be left empty. Upon pressing the *Confirm* button, the program checks if the database name

contains valid characters. If yes, it creates the database in the `\Databases` folder and selects it immediately.

4.3.3 Select data provider

A data provider can be chosen from the drop down box in a Group box called *Select Data Provider*. This drop down box is filled with names of libraries from the folder *DataProviders*, that are implementing the IAnalyzer interface. Clicking on the *Details* button displays a brief description of the selected provider. For example, the *GoogleProvider* library provides only one year old historical stock exchange data. Again, the *Confirm* button sets up the chosen provider as the application default.

4.3.4 Set up initial date for downloading

Unlike Yahoo, Google does not provide historical data older than one year. The user can set up a date representing the starting point for historical data. This tells the provider to ignore older records. Setting the date can be done by clicking on the date-time picker and selecting a date.

4.4 Add or delete assets

The user can add or delete assets by clicking on the *Assets* label in the Menu, and selecting the desired operation. There are two different forms - Add Asset form (Fig. 4.3(a)) and Delete Assets form (Fig. 4.3(b)). To add a new asset, the user needs to possess the unique symbol under which the asset represented on the market. The name can be filled randomly. By clicking on the *Add* button, the application tries to download the historical data and add the asset to the database. If an error occurs, a notification is displayed on the screen.

Deleting assets is as simple as selecting their names in the *Delete Assets Form*, and clicking on the *Delete* button. All of the records belonging to the selected assets will be removed from the database.

4.5 Chart

In this section we discuss how to draw a chart and work with it.

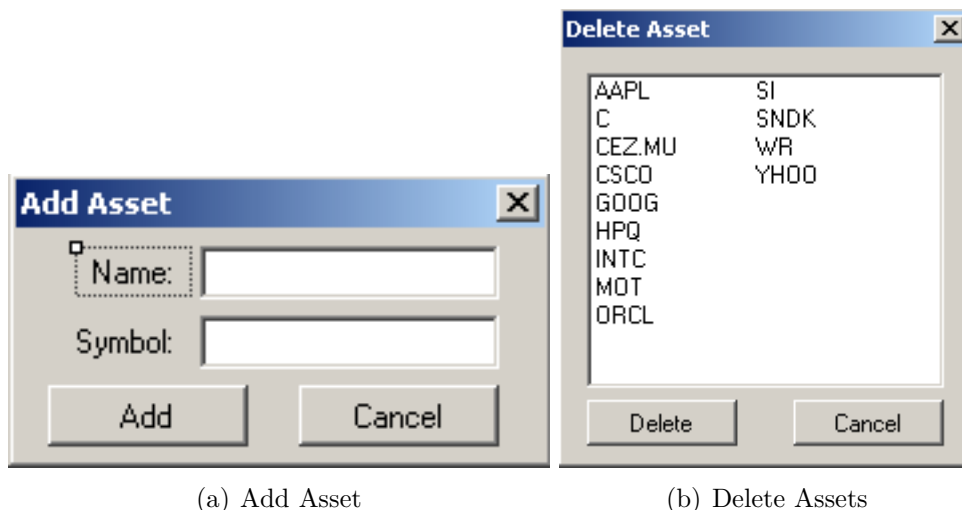


Figure 4.3: Add and delete assets

4.5.1 Display chart

The chart of the instrument is drawn by clicking on the name of the instrument or on its symbol in the top left list view. The application loads the data from the database, filters them and displays them into the chart control (see 4.1). Two default charts are displayed representing instruments price and volume.

4.5.2 Operate with chart

The figures 4.4 represent controls for changing the chart appearance and group values. The instrument value chart can be represented as:

- **Line** chart, showing the opening price
- **Bar** chart, representing the records as an Open-High-Low-Close bar (see Fig. 2.3(b))
- **Candlestick** chart, representing the records as in the Bar chart but in a different format (see Fig. 2.3(a))

The values can be grouped in the following way:

- **B1** no grouping, only display the data from the database

- **W1** grouping the values into the week interval. For the line chart, this calculates the average of the closing prices. For OHLC and Candlestick, this calculates the average for open and close prices. Takes the maximum value from the highs and minimum value from the lows. Calculates the average for the indicators' values
- **M1** grouping the values into the month interval in the same way as *W*.

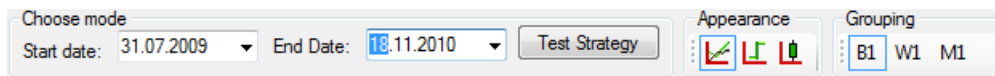


Figure 4.4: Set up a simple strategy

4.5.3 Zooming and scrolling

The chart can be zoomed in and out by using the + and - keys or by using the mouse wheel. The chart is reset to its initial view by pressing the small circular button on the left side on the Y-axis scrollbar and over the X-axis scrollbar. We can zoom a specific part of the graph by clicking and dragging the mouse. The actual mouse selection is shown in light gray color.

4.6 Apply indicator

The list of available indicators shows the loaded and linked indicators. The application supports 3 different chart areas, with more than one indicator in the same chart. The indicator is calculated and inserted into the chart by clicking on its name. The indicators will not be calculated and displayed if there is no instrument selected. Before the calculation of the indicator, the application opens a form with the settings if the indicator needs them. For example MA needs to set up time frame and shift, but On Balance Volume does not need any additional attributes, and is calculated and displayed immediately. If the user wants to change indicator settings, he has to click on it again to unload it from the chart and click again to repeat the procedure.

4.7 Apply simple strategy

To apply a simple strategy an instrument's chart has to be drawn. The user then clicks on the *Strategies* \Rightarrow *Simple*. It opens the *frmCreateStrategy* form (Fig. 4.5) where the strategy can be set up. The user can choose if he want to

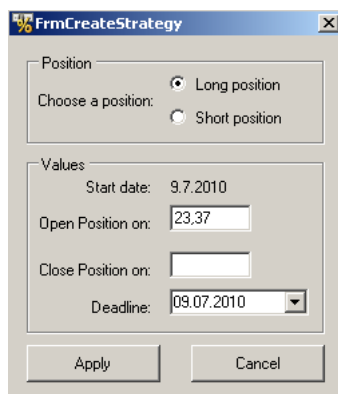


Figure 4.5: Set up a simple strategy

enter a *Long* or *Short* position. In the *Value* group box, he is informed about the start date. The *Open Position on* value holds the start dates' opening price, but it can be changed. The closing price and the *Deadline* date, which tell the application about the last possible date to close the position, also need to be filled. Clicking the *Apply* button starts evaluation of the strategy. The application then displays the results and eventually marks in the chart the opening and closing position.

4.8 Create and run strategy scripts

4.8.1 Create a strategy

The Chapter 3 discusses the strategy scripts' background. The scripts template is shown in the **Appendix A**. The script has to implement the *IStrategy* interface and *MarshalByRefObject*. The user has to set the fee amount for the transaction. To simplify his work, he can sets the constants. Method *RequestDemands()* enables user to set the fee, request data and set the value, which will be used for Open or Close the position. The *Evaluate(Dictionary*

`<string, double> value`) method represents the place to implement strategy logic. The user receives requested data in an array indexed by string for transparent representation of the particular value. The user can call `host.OpenPosition(direction)`, where `direction` means Buy or Sell. The position can be closed by the methods or by the script itself. The methods `Stop loss`, `Trailing Stop loss`, `Take profit` are implemented. The description how to use them is specified in the **Appendix A**. Method `host.Trailing(value)` in a `if` condition enables this function to close position, same as for `Stop loss` and `Take profit`.

4.8.2 Create and run strategy scripts

Chapter 3 discusses the strategy scripts' background. There are two ways of loading a script. The first is by clicking on the *Strategies* \Rightarrow *Scripts* (Fig. 4.6). By clicking on the *Browse* button, the file input dialog will be shown and

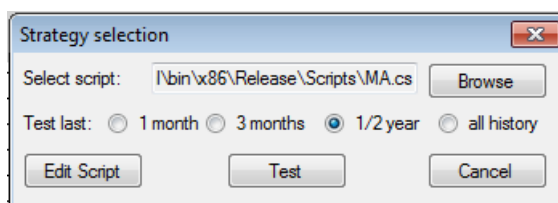


Figure 4.6: Select a script

the script can be selected for evaluation. The default folder for the scripts is `em \Scripts` but the user can browse to another directory. After the selection has been made, the script can be edited by clicking the *Edit Script* button. It will start the default program for creating and editing the `*.cs` files on the user's computer. The *Test* button starts evaluating the script. The second way of loading a script is by clicking on the *Test Strategy* button. The button opens the same form as in the previous example. The script eventually opens the indicators' settings form. If no error has occurred the table with results (example in Fig. 4.7) is displayed. The table contains all information about the values and events which caused opening and closing of the positions. Under the table is the summary of the strategy, which displays the total strategy profit, the number of Long/Short positions which gained profit or created a loss, and the overall profit and loss for the Long/Short positions. The application marks these events into the chart. The user can export the

results into CSV or Excel files by selecting these options and by clicking the *Export* button. The user can change the name of the file to the description of the strategy. The files are exported into *\Output* folder in the main directory of the program.

The screenshot shows a window titled "Strategy results" containing a table of trade events, a summary section, and an export options section.

Time	Position	Event	Price	Profit	SimpleMA	Details
16.6.2009	Sell	Close	420.5	1	427.8	TakeProfit event
29.6.2009	Buy	Open	426		418,6	
30.6.2009	Buy	Close	424	-2	417,77	Trailing StopLoss event
2.7.2009	Sell	Open	415,41		416,15	
6.7.2009	Sell	Close	414,41	1	415,36	TakeProfit event
13.7.2009	Buy	Open	416,17		412,84	
3.2.2010	Buy	Close	533,87	117,58	555,86	Trailing StopLoss event
17.8.2009	Sell	Open	451,5		452,15	
18.8.2009	Sell	Close	450,5	1	452,76	TakeProfit event
31.8.2009	Sell	Open	459,79		460,09	
2.9.2009	Sell	Close	458,79	1	459,59	TakeProfit event
9.9.2009	Sell	Open	459,06		462,61	
10.9.2009	Sell	Close	460,06	-1	463,51	StopLoss event
2.10.2009	Sell	Open	483,74		492,68	
5.10.2009	Sell	Close	484,74	-1	493,52	StopLoss event
29.10.2009	Sell	Open	543,01		544,29	
30.10.2009	Sell	Close	544,01	-1	545,67	StopLoss event
2.11.2009	Sell	Open	537,08		546,41	
3.11.2009	Sell	Close	536,08	1	546,39	TakeProfit event
27.11.2009	Sell	Open	572		573,53	
30.11.2009	Sell	Close	573	-1	575,16	StopLoss event
7.1.2010	Sell	Open	608,4		612,22	

Strategy result summary
Profit taken: 604,32

Long positions:66 Successful :26 Fail:40 Profit:683,52 Loss:-69,2 Total:614,32

Short positions:100 Successful :45 Fail:55 Profit:45 Loss:-55 Total:-10

Export Results
Filename: GOOG_10_12_2010 CSV (comma delimited) (*.csv) Excel (*.xls) Export

Ok

Figure 4.7: Example of a strategy result

Chapter 5

Conclusion

The program provides stock exchange data from the remote server only by specifying the instrument's symbol and creates an extendable platform for plug-in based indicators and data provider. If a new subject appears on the market, which can provide more frequent data with less latency, the user can simply create a new class library which downloads this new data, and add it to the program without additional changes in the code. The new indicators, can be easily distributed among users. Scripts and databases are portable and not dependent on the user's locale settings. We have also managed to maximize the benefits of the chart control, which is very flexible, but can get slow for large sets of data. We have also created instruments to evaluate strategies by transforming them into scripts. We have made a background support for creating and running scripts. One of the areas that can be improved in the future is strategy script writing, which could be done by using pseudocode which is closer to natural language.

5.1 Comparison with another available implementation

On the market there are various business application designed for trading in real time on the stock exchange. Some of them, like XTB or MetaTrader focus on the novice users. They provide robust platforms for trading with fictional money, making the beginnings more attractive for users since they can immediately experience the actual trading environment. Still this robustness can bring confusion to the novice by distracting him from the un-

derlying principles of trading. This program is aimed exactly at those users who want to focus on the basics and slowly build their way up to commercial trading software. Our program tries to simplify and filter only the necessary tools for creating and testing strategies and as such can find its place in the crowded space of educational trading software.

Bibliography

- [1] Wikipedia. *Wikipedia [online]*. 2010 [cit. 2010-12-02]. Dividend. Available on WWW: <http://en.wikipedia.org/wiki/Dividend>.
- [2] Ross Joe. *Trading By The Book* 5 edition. USA : Ross Trading, 1985. How To Get Wisdom, p. 10-11. ISBN 976810824X
- [3] DALTON, James F.; JONES, Eric T.; DALTON, Robert Bevan. *Power Trading With Market Generated Information*. 2 edition. USA : Traders Press, 1999. Introduction, p. 1-6
- [4] ARUN, Arun. *Market Trends [online]*. 2009-02-07 [cit. 2010-12-02]. Stock Market Position, <http://weblogsurf.com/february-2009-stock-market-position>.
- [5] Wikipedia. *Wikipedia [online]*. 2010 [cit. 2010-12-02]. Trh (ekonomie). Available on WWW: [http://cs.wikipedia.org/wiki/Trh_\(ekonomie\)](http://cs.wikipedia.org/wiki/Trh_(ekonomie)).
- [6] Wikipedia. *Wikipedia [online]*. 2010 [cit. 2010-12-02]. Stock valuation. Available on WWW: http://en.wikipedia.org/wiki/Stock_valuation.
- [7] O'Neill, Michael. *BUS 106 [online]*. 2010 [cit. 2010-12-02]. Canadian Market Economy - Demand and Supply, http://www.mhoneill.com/106/chap/ch1/ch1_c.html
- [8] Wikipedia. *Wikipedia [online]*. 2010 [cit. 2010-12-02]. Burza. Available on WWW: <http://cs.wikipedia.org/wiki/Burza>.
- [9] Wikipedia. *Wikipedia [online]*. 2010 [cit. 2010-12-02]. Fundamental analysis. Available on WWW: http://en.wikipedia.org/wiki/Fundamental_analysis.

- [10] Wikipedia. *Wikipedia [online]*. 2010 [cit. 2010-12-02]. Psychologická analýza. Available on WWW: http://cs.wikipedia.org/wiki/Psychologicka_analyza.
- [11] Wikipedia. *Wikipedia [online]*. 2010 [cit. 2010-12-02]. Technical analysis. Available on WWW: http://en.wikipedia.org/wiki/Technical_analysis.
- [12] moneyterms.co.uk. *moneyterms.co.uk [online]*. 2010 [cit. 2010-12-02]. Technical analysis. Available on WWW: <http://moneyterms.co.uk/technical-analysis/>.
- [13] LANGAGER, Chad; MURPHY, Casey. *Investopedia [online]*. [cit. 2010-07-09]. Analyzing Chart Pattern: Head and Shoulders, <http://www.investopedia.com/university/charts/charts2.asp>
- [14] LANGAGER, Chad; MURPHY, Casey. *Investopedia [online]*. [cit. 2010-07-09]. Analyzing Chart Patterns: Double Top And Double Bottom, <http://www.investopedia.com/university/charts/charts4.asp>
- [15] Investopedia. *Investopedia [online]*. 2010 [cit. 2010-12-02]. Technical Indicator. Available on WWW: <http://www.investopedia.com/terms/t/technicalindicator.asp>.
- [16] moneyterms.co.uk. *moneyterms.co.uk [online]*. 2010 [cit. 2010-12-02]. Momentum Investing. Available on WWW: <http://moneyterms.co.uk/momentum-investing/>.
- [17] moneyterms.co.uk. *moneyterms.co.uk [online]*. 2010 [cit. 2010-12-02]. Growth Investing. Available on WWW: http://moneyterms.co.uk/growth_investing/.
- [18] moneyterms.co.uk. *moneyterms.co.uk [online]*. 2010 [cit. 2010-12-02]. Value Investing. Available on WWW: http://moneyterms.co.uk/value_investing/.
- [19] Microsoft SQL Compact edition 3.5, <http://www.microsoft.com/Sqlserver/2005/en/us/compact.aspx>.
- [20] MSChart Control for .NET 3.5, [http://msdn.microsoft.com/en-us/library/3ks53324\(VS.71\).aspx](http://msdn.microsoft.com/en-us/library/3ks53324(VS.71).aspx).

- [21] Wikipedia. *Wikipedia [online]*. 2010 [cit. 2010-12-02]. Moving average. Available on WWW: http://en.wikipedia.org/wiki/Moving_average.
- [22] Wikipedia. *Wikipedia [online]*. 2010 [cit. 2010-12-02]. On-balance volume. Available on WWW: http://en.wikipedia.org/wiki/On-balance_volume.
- [23] Wikipedia. *Wikipedia [online]*. 2010 [cit. 2010-12-02]. Relative Strength Index. Available on WWW: http://en.wikipedia.org/wiki/Relative_Strength_Index.
- [24] CS-Script - The C Sharp Script Engine, <http://www.csscript.net>
- [25] Investopedia. *Investopedia [online]*. 2010 [cit. 2010-12-02]. Stop-Loss. Available on WWW: <http://www.investopedia.com/terms/s/stop-lossorder.asp>.
- [26] Investopedia. *Investopedia [online]*. 2010 [cit. 2010-12-02]. Trailing Stop. Available on WWW: <http://www.investopedia.com/terms/t/trailingstop.asp>.
- [27] Investopedia. *Investopedia [online]*. 2010 [cit. 2010-12-02]. Take-Profit Order - T/P. Available on WWW: <http://www.investopedia.com/terms/t/take-profitorder.asp>.

Appendix A

Scripts' template

```
using System;
using System.Data;
using System.Collections.Generic;
using StrategyInterface;
using StockEval;

class Script : MarshalByRefObject, IStrategy
{
//This part cannot be changed and it is used to inform
//the script with whom it can communicate
    IStrategyHost host = null;
    public IStrategyHost Host
    {
        get
        {
            return host;
        }
        set
        {
            host = value;
        }
    }
//set the fee amount for the transaction
    double fee = 0.1;
}
```

```

//These constants simplify user's work with data
private const string OPEN = "OpenPrice";
private const string CLOSE = "ClosePrice";
private const string BUY = "Buy";
private const string SELL = "Sell";
//Sets the values for Stop Loss, Take Profit and Trailing stop
//loss function
//Can be changed or the values can be manually inserted into
//the function calls
private const double RISK = 2;
/// <value>
/// Request the data from the application
/// </value>
public void RequestDemands()
{
    //Set the fee for the transaction
    host.Fee = fee;
    //Inform the application about which values we want
    host.Demands(OPEN, CLOSE, "RSI");
    //Set the value that will be used for Open or Close the position
    host.PriceColumn = OPEN;
}
/// <summary>
/// Evaluating function for the script, this function is
///called from the application
/// The logic of the script must be here
/// </summary>
/// <param name="value">Collection of values, which were
requested in the previous step</param>
public void Evaluate(Dictionary<string, double> value)
{
    //Some conditions for opening the position
    //Functions (Take Profit, Stop Loss) have to be set when
    //opening the position
    if (value["RSI"] > 70)
    {
        //Example of using StopLoss function
        //SetStopLoss initializes the function and needs to have 2 values -

```

```

//StopLoss value and the specification of the position
host.SetStopLoss(value[OPEN] + RISK, SELL);
//Same as StopLoss
//Example of function call without constants:
//host.SetTakeProfit(value[OPEN] - 12.50, "Sell");
host.SetTakeProfit(value[OPEN] - RISK, SELL);
//Open a short position
host.OpenPosition(SELL);
}
//Condition for opening a long position
if (value["RSI"] < 30)
{
//This function enables the user to work with TrailingStopLoss
//It has 4 parameters:
// 1.Initial Open price of the position
// 2.Stop Loss value
// 3.First value when the Stop Loss has to be changed
// 4.Percentage of the gain added to the Stop Loss
// 5.Specification of the position we are in
host.SetTrailing(value[OPEN], RISK - 1, RISK, 70, BUY);
//Open a Long position
host.OpenPosition(BUY);
}
//Condition to test if the TakeProfit function for this value
//is activated or not
//Eventually close the position
if (host.TakeProfit(value[OPEN]))
{
//Close the position
host.ClosePosition();
}
//Condition to test if the Trailing Stop Loss function
//for this value is activated or not
if (host.Trailing(value[OPEN]))
{
//Close the position
host.ClosePosition();
}
}

```

```
//Condition to test if the StopLoss function for this value
//is activated or not
if (host.StopLoss(value[OPEN]))
{
    //Close the position
    host.ClosePosition();
}
}
```


Appendix B

CD contents

The accompanying CD has the following structure:

- *\Txt*
 - *\Latex* - Latex source files with used images
 - *Thesis.pdf*
- *\Setup*
 - *Setup.exe*
 - *StockEval.msi*
- *\Doc*
 - *StockEval.CHM* - documentation generated by the Sandcastle
- *\Src*
 - *StockEval* - Visual studio 2008 solution with all source files