Currency Trading System Development

An Interactive Qualifying Project Report
Submitted to the Faculty of
WORCESTER POLYTECHNIC INSTITUTE
In Partial fulfillment of the requirements for the
Degree of Bachelor of Science

WPI

By
Jake Barefoot [REL 1805]
Michael Cullen [REL 1803]
Aung Khant Min [REL 1807]

Submitted to
Professor Reinhold Ludwig
Professor Gbetonmasse Somasse

Date: 30 April, 2019

Abstract

In a rapidly evolving world of market dynamics, it has become increasingly difficult to consistently gain an investment edge. The goal of this IQP is to develop an automated trading system in the Metatrader platform. The project includes three individually developed algorithmic trading systems and one combined system that merges the three individually developed systems. Each system is optimized over a number of recent years in order to maximize the return on investment. The approach allows the combined system to be a broad consolidation of the individual system performances.

Table of Contents

1. Introduction

1.1 Problem Statement

The recent widespread use of the internet has drastically changed the financial industry landscape for investors. The average person now has unprecedented access to a vast amount of information that can be used to gain wealth. With the ease and speed of data access, any potential investor can access financial reports, market indicators, and press releases for any company, currency, or commodity.

The development of the internet has also led to drastic changes for brokers. The people who facilitate the transactions traditionally had high commission rates, because the client was paying for access. The tech boom caused an increase in the amount of brokers leading to ultra competitive commission rates. Brokers now don't even need to be human. Online brokers are now a viable way to invest in financial markets. The commission rates are very low, and the transactions can be completed in milliseconds.

The amount of new participants in the financial world create a hyper competitive space, where a prospective trader needs to have a novel approach in order to achieve success. Developing a novel system requires research on what has been achieved in the past, and what factors affect the price of an asset. Personal traders can develop systems that rely on price data, financial reports, consumer sentiment, and up to date news. Traditional money managers are still a viable option, but achieving financial independence can be done with a solid understanding of what changes the price of a stock, currency, bond, option, etc. Once a trader has developed their

system it needs to be tested to determine its viability. The problem for traders is implementing a strategy in a way that can be consistently recreated, and generate useful results to help understand if the trading system is worth pursuing.

1.2 Project Description

The goal of our project was to create an integrated system of three individual automated currency trading systems that can be used as a baseline for future trading. The advantages of using automated trading systems include the ability to evaluate performance on historical data, and to limit the introduction of human bias in individual trades. To develop the systems, we researched the structure of the foreign exchange market, trading styles, and analytical methodologies used in the field. We then chose a trading platform and learned the coding language to develop and test three individual systems. Lastly, the systems were integrated into one consolidated “system-of-systems” in an attempt to outperform the individual systems. The results of each approach were recorded and analyzed.

2. Understanding The Currency Market

The first step in reaching the goals of this project was to develop a detailed understanding of the currency market, as well as understand the options of gaining wealth in financial markets.

2.1 Trading vs Investing

The two broad philosophies to gain wealth in financial markets are investing and trading. Investors attempt to build wealth by holding financial assets for an extended period of time. The underlying reasoning is that markets usually mature over time, so an investor will buy stocks then reinvest profits to continue gaining wealth. The time frame for investors can vary, but certain assets can be held for decades if the outlook is promising. Investors will generally attempt to look at investment factors that will affect long term performance for the asset such as the management structure, profitability, long-term company goals, trade balances, or political sentiment. There are many types of investing such as value investing which looks at lower price yet well performing assets, and growth investing which looks at emerging assets with potential, but they all share the same goal of having assets grow over time (Chan, 2004).

Traders employ a different method to gain wealth. Traders look for ways to buy a stock, or currency, and sell quickly when there is an imbalance in the price of an asset. Traders look to make smaller profits in the short term that will add up to more than what can be made by just holding the asset. Traders can make money on both sides of the market by buying on an uptrend and selling on a downtrend. The time frame for traders is shorter than investing, but can vary significantly. Some traders keep positions for weeks or months, and some only hold positions for

a few seconds. Successful traders look at various technical indicators that indicate short term performance such as, oscillators, moving averages, and others that will be discussed in Chapter 3.

It is important to understand two significant differences between investing and trading before participating in financial markets. The first difference is the higher time commitment for trading. In order to be a successful trader, positions can be opened and closed on a daily basis. This means a trader needs to be vigilant with their positions, while an investor should monitor their positions less frequently and focus on long term growth. The time commitment for trading can be mitigated with an automated system. Another major difference is risk. Investing involves possibly smaller payoffs, with lower risk than trading. Most financial markets see a consistent growth year to year, meaning an investor benefits from that growth by constantly staying in the market. A trader can earn a significantly larger percentage, but relies on less predictable short term movements which increases risk significantly. These factors can help a new financial prospector determine which portfolio management strategy suits their financial situation and personality.

2.2 Currency markets

Forex is the global market that allows the exchange of one currency for another. The market facilitates transactions that allow the investor to take advantage of the varying exchange rates of the different currencies. If used correctly the $5 trillion forex market can be a very profitable venture (Forex Market Overview, 2019). In currency markets, the dominant earning strategy is trading. This is for the simple reason that currencies do not have the same upward

trend as stock trading. For example, If you traded 100 US Dollars for Euros in 1994, those Euros you purchased would be worth the same 100 US Dollars in 2017. To understand how money can be made in the currency market it is important to know what groups are involved in the forex market.

2.2.1 Who is involved

Forex is unique because it is a decentralized market where anyone can participate. This decentralization allows for different banks, and brokers, to offer different prices for the currency. Even though currency quotes are not fixed, brokers almost always offer the best price because of the extremely high competition. Even with the openness of the forex market there is still a hierarchy for trading. The super banks are at the top of the market. These large banks set the spreads and make a massive amount of trades per day. At the bottom of the ladder are the speculators. Speculators comprise 90% of the trading volume and are comprised of individuals with varying amounts of currency (Forex Market Overview, 2019). Anyone can participate in the forex market with as little as $50 in an account.

2.2.2 Advantages and disadvantages

Table 1 highlights some of the advantages and disadvantages of creating a trading system for the forex market.

Table 1: Advantages and Disadvantages of the Forex Market

Advantages	Disadvantages
Little/No Commissions	Increased Risk with Leverage
No Fixed Lot Sizes	Low Volatility
Hard to Influence the Market	Government Intervention
Less Volatile	
24-Hour Trading	
Leverage	

There are many advantages to trade in the forex market. One advantage is there are no commissions because most brokers make money through the 'bid/ask' spread. Forex is great for smaller investors for a number of reasons. There are no fixed lot sizes, meaning an investor can spend as little as $50 to make an account. It is also difficult for a single institution to heavily influence the market because of how much currency is traded. For smaller investors this means less volatility, and more security for investments. The last key advantage for forex trading is all day activity. There are different markets all around the world that are constantly trading currency, so investors with day jobs can trade during their off hours. In the stock world, the popular markets have a more narrow trading time frame.

With all these advantages there are some disadvantages. One possible disadvantage would be the increased risk when trading with high leverage. Leverage is when you are given the buying power more than the value of your account. This allows you to buy more currency than you can afford. The result could mean big profits, or draining an account to zero. Forex markets allow the user higher leverage than stocks because of the fixed profit that brokers can make.

Another disadvantage would be low volatility. Currency pairs only shift a few hundredths or tenths of a percentage point in a day. This little price movement can make it difficult for someone without large capital to see any real profits without using leverage. Another disadvantage would be intervention by governments. The forex market is not fully governed by the free market, the currencies are controlled by governments that will have the option to raise interest rates, or transfer funds.

2.2.3 Currency pairs

This massive market of global currencies primarily trades eight major currency pairs. Any currency trading in the forex market must be done in pairs because the investor is essentially selling one currency to buy another. There are seven major pairs that trade the US Dollar (USD) and many minor crosses that trade various other currencies. The major forex currency pairs are shown below in Table 2.

Table 2: Seven Major Currency Pairs (Major Currency Pairs, 2017)

Currency Pair	Countries
EUR/USD	Euro zone / United States
USD/JPY	United States / Japan
GBP/USD	United Kingdom / United States
USD/CHF	United States/ Switzerland
USD/CAD	United States / Canada
AUD/USD	Australia / United States
NZD/USD	New Zealand / United States

The currency pair is represented by two three letter symbols of the currency. The first is the base currency and the second is the quote currency. The base currency is is how much of the quote currency is needed to get one unit of base currency. For example, a EUR/USD is 1.5. This means you would have to give 1.5 USD to get 1 EUR.

2.2.4 Trading times and markets

As mentioned before, the forex market is open all day during the week, however there are different markets open at different times. For example, New York is open 8:00 AM – 5:00 PM, and London is open 3:00 AM – 11:00 AM. There is currency moving all hours of the day, but the most active sessions are London and New York. There is also higher trade activity during market overlap times. This means the best time to trade currency on the east coast is from 8:00 AM – 11:00 AM when the New York and London markets are active. It is important to note that markets have higher liquidity during the middle of the week. Table 3 below shows the session times for popular forex markets.

Table 3: Forex Trading Times (Forex trading sessions table, 2017)

Time Zone	EDT (April – October)	EST (October – April)	GMT
Sydney Open	6:00 PM	4:00 PM	10:00 PM
Sydney Close	3:00 AM	1:00 AM	7:00 AM
Tokyo Open	7:00 PM	6:00 PM	11:00 PM
Tokyo Close	4:00 AM	3:00 AM	8:00 AM
London Open	3:00 AM	3:00 AM	7:00 AM
London Close	12:00 PM	12:00 PM	4:00 PM
New York Open	8:00 AM	8:00 AM	12:00 PM
New York Close	5:00 PM	5:00 PM	9:00 PM

2.2.5 Pips/Lots

A percentage increment point, or pip, is the smallest unit of measurement to express the change in value between two currencies. Because there is not much volatility in currency markets, a pip is usually the last decimal place in the quote and is usually four decimal places out for most currencies, and two places out for Japan. Knowing the value of a pip determines how much money an investor made from a position. The formula for calculating pip value is shown in Equation 1.

(Value change in counter currency) x (Exchange rate ratio) = pip value in base currency (Eq. 1)

It is important to know how pip values are calculated, but any electronic trading platform will calculate pip value, and total trade profit, for the user.

It is also vital to know that currency is traded in lots. It is very unusual, and impractical to just buy a couple units of currency, because there will be no realizable profit with normal currency movement. A standard lot is 100,000 units of currency. Most trading platforms will allow the user to trade smaller sizes such as 10,000 units, or even 100 units. Leverage can also be used to trade with larger lot sizes than your account would normally be able to purchase, but as discussed previously this requires increased risk.

2.2.6 What affects currency rates

There are many factors that can change a currency ratio. The main four would be:

- Interest rates - Higher interest rates mean higher increase in value for foreign investments. If the interest rate is high compared to another country the currency value increases. Vice versa for low interest rates (Froot, 1990).

- Inflation - purchasing power of a currency is equally important as interest rates. As inflation increases the amount of goods purchasable with a currency decreases, and attractiveness decreases for foreign investors, vice versa for decreases in inflation where the real value of a currency increases (Froot, 1990).
- Trade balance - difference of value for exports and imports. A positive trade balance for a country implies desire for the country's goods and foreign countries are forced to exchange more currency for domestic currency. As a result the currency value increases with positive trade balance and currency value decreases with a deficit (Fischer, 1993).
- Government debt - debt owed by the central government. Higher government debt can lead to inflation with less purchasing power for a currency. Inflation will cause investors to leave. Higher government debt generally leads to a decrease in currency value (Patel, 2014).

There are other factors that could change the currency ratio such as trade terms between countries, political changes, unemployment, natural disasters, gross domestic product, etc. These factors are backed by macroeconomic theory and are responsible for a large portion of forex movement, but there are also changes cause by the speculators mentioned above. Some price movements caused by the currency traders are in response to other technical analysis tools discussed in Chapter 3.

3. Trading Systems

This Chapter explores the concept of trading systems, how to develop a trading system, and the factors that contribute to a successful system.

3.1 What is a Trading System?

A trading system is an established set of rules which are used to consistently make trades that maximize profit and limit risk. A trading system will have entry and exit conditions based on technical, fundamental, or analytical tools. Because of the amount of traders participating in the forex market, traders need to develop an edge with their system. An edge would be a novel set of rules and analytical tools that help to overcome the somewhat random movement of a market. In theory, the market price is completely random, but in reality it is determined by a large number of factors including human emotion. A trader with an edge in the market will be able to exploit unique non-random factors that produces a positive outcome in a somewhat random market.

3.2 Trading platforms

There are a lot of trading platforms to choose from such as Ally Invest, TradeStation, TD Ameritrade, Interactive Brokers, Charles Schwab, EOption and Metatrader. Trading platforms are applications that allow users to look at price, or fundamental data for different assets. Typically trading platforms will offer the ability to make manual trades or allow the user to develop automated trading strategies. For this project, the focus will be on Metatrader as it is the largest forex broker, and allows for automatic trading system development.

3.2.1 Metatrader

Metatrader has been adopted by the forex market as the standard market trading platform. Hundreds of brokers around the world have been using it since November 2000 (MetaQuotes, 2019). Metatrader offers a stable platform with enhanced security so that the trader can open and close forex positions without worry. Also, the Metatrader platform provides advanced technology for its users. For instance, it has over fifty technical indicators pre-installed. The user can choose from nine different time frames and three different chart times in order to fit their own trading style.

3.3 Trading styles

Trading support and resistance levels can be divided in two ideas: the bounce and the break. When playing the bounce, it is wise to tilt the odds in the favor of the trader and find some sort of confirmation that the support or resistance will hold instead of setting the entry order right on the line. When the break occurs, there are two ways to go at it: the aggressive way and the conservative way. The aggressive way is to buy or sell whenever price passes convincingly through a support or resistance zone. The conservative way is to hold on to the trade, wait for price to make a “pullback” to the broken support or resistance level and enter after the price bounces.

3.4 Exit conditions

A successful forex trader also needs to address exit conditions for an order. The first exit condition is a stop loss. A stop loss is a price point that that is set to limit the potential loss of an

order. A stop loss is initially set in the opposite direction of an order, in case of a price shift that would negatively affect the return. There are many strategies that can be used to set stop losses that maximize return. A basic option would be to use a fixed stop loss. An example of a fixed stop loss would be setting a stop loss 20 pips below the price of a currency. This basic strategy limits risk, but stop losses can be made more robust. Another stop loss strategy would be to place a stop loss based on certain trends in price. The placement could be made based on resistance levels, range of recent prices, or recent pull backs (Grimes, 2012). A third strategy could be to have a trailing stop loss that can increase or decrease based on trends. This might mean setting a stop loss, having an order initially succeed causing the stop loss to rise above the take profit point. Stop loss strategies that are based on price movement help to further limit risk of an order.

Another exit condition for an order is a take profit. A take profit is a price that is set as a point to exit the order that results in a profit. A take profit is used useful to a trader to limit the risk of continuously staying in the market. A basic strategy would be to have a fixed take profit. This carries the dame disadvantages as a fixed stop loss, but is more valuable than no profitable exit condition. Other logical take profit strategies could include setting an exit at resistance or support levels, a defined risk/reward level, or specific time.

A trader does not have to set stop losses, or take profits when creating an order, but exit conditions do have to be set. The specific values do not have to be given when opening a trade with Metatrader, or any other trading platform. A trader can choose to exit the market based on market indicators that will be discussed later. Analyzing market indicators can help to increase chances of determining price movement. No matter what exit conditions a trader chooses, the

rules needed to be followed diligently to increase consistency and reliability of the trading system.

3.5 Types of analysis

There are three basic types of forex market analysis: Technical analysis, fundamental analysis, and sentiment analysis. Technical analysis is a very subjective way of looking at historical price movements and determining the current trading conditions and potential price movements. Fundamental analysis looks at factors such as economic, social, and political forces that could potentially affect the supply and demand of the product. Sentiment analysis is using a trader's own opinion or understanding of the market to either follow it or go against it.

3.6 Technical analysis indicators

There are a multitude of technical analysis indicators that are used to try to predict the future price of a commodity. These indicators can also help a trader determine entry and exit conditions. Some common technical indicators used in forex markets are described below.

3.6.1 Moving Averages

A moving average is an indicator used in technical analysis that helps smooth out price action by filtering the noise from the price (Grimes, 2012). The two most common forms of moving averages are simple or exponential. Simple moving averages are calculated by averaging the previous x closing prices. Exponential moving averages have a more difficult formula that still averages the previous x periods, but adds weight to more recent values. Adding more price

periods to these averages smooths the curve, but these longer averages are more prone to market lag.

One of the most common ways to utilize moving averages would be to plot a shorter moving average and a longer moving average on the same plot. The shorter moving average would react quickly to the market price, while the longer average would show general trends. Investors then determine when to buy or sell based on the crosses of these two lines . The first cross would be when the shorter average crosses above the longer moving average. This would signal that the market is going to rise so an investor would want to buy. The second cross would be when the shorter average crosses below the longer average. This would signal that the market is going to fall, so an investor would want to sell. There are many ways to utilize moving averages by adding lines, or changing the length of the averages. The versatility of moving averages have made them an extremely popular economic indicator.

3.6.2 Moving Average Convergence Divergence

The moving average convergence divergence (MACD) indicator is a momentum indicator that shows the relationship between two moving averages of price. The value is determined by subtracting a longer moving average from a shorter moving average. Then a third average is used as a signal line that help determine when to buy and sell. Different investment platforms allow the user to select different ranges for three averages, but no matter the amount of periods, the investment strategy remains the same. Shown in Figure 1 is an example of MACD.

Figure 1: MACD (Moving Average, 2019)

There are three patterns that come from the MACD that can signal an investor to buy or sell. The convergence pattern is when the two averages cross. The implications of these crosses are similar to the crosses described in the moving averages section. The next pattern is divergence. Divergence is when the market price moves away from the MACD. One example would be if the current price is rising, but the MACD is falling. In this case, the investor would want to sell because the market price is predicted to fall. On the other hand, if the market price is falling and the MACD is rising, the prediction is the market price will rise in the future. The third pattern would be a dramatic rise in the MACD. If this happens, the shorter average has vastly increased compared to the longer average. The prediction would be the market price has been overbought and will return to a more normal level. In this case, an investor would want to short the market until the market price levels out (Grimes, 2012). The MACD is a very popular indicator because its historical accuracy and graphical intuitiveness.

3.6.3 Relative Strength Index

The relative strength index is a momentum indicator that measures the speed and change of price movements. The RSI uses trends of price changes to determine a "score" of market price that ranges from 0 to 100 (Bauer, 1999). A score of 100 signifies an extreme bull market where the market is constantly rising. A score of 0 signifies an extreme bear market where the market is constantly falling. Investors tend to use the scores of 30 and 70 to determine when to buy or sell. An example of RSI is shown by the blue line in Figure 2.

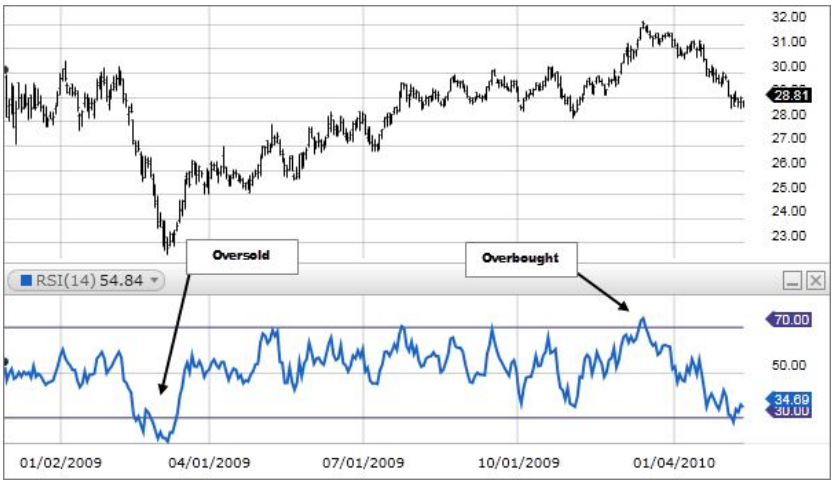


Figure 2: RSI (Relative Strength Index, 2019)

An RSI of 70 would mean the market is overbought. When a market is overbought the prediction would be the market price is about to fall in the future. The conclusion for the investor would be to short the market. An RSI of 30 would mean the market is oversold. When a market is oversold the prediction would be the market price is about to rise in the future. The conclusion for an investor would be to long the market. Because RSI is a relative measure, the exact levels of overbought and oversold can be altered depending on investment strategy.

3.6.4 Stochastic Indicator

The stochastic indicator is a momentum indicator that shows the location of the close price relative to the high-low range over a number of periods (Bauer, 1999).The highest high and lowest low are determined by the price over the defined number of periods. The user and investment strategy determines the time frame used. An investor can measure the price of a market over hours, or days, with longer time periods being used for more conservative investors. The stochastic indicator does not follow price or volume of a market, it follows the momentum of the price. The thought behind following the momentum of price is that it will change before the actual price does. Figure 3 shows an example of the Stochastic indicator in red.

Figure 3: Stochastic Indicator (Double Smoothed Stochastic, 2019)

Much like RSI, the stochastic indicator ranges from 0 to 100, and set levels help determine when to buy or sell. Because both indicators are bounded, the investment logic is very similar. When a market is overbought an investor should buy, and when a market is oversold an investor should sell. Although the overbought and oversold levels vary, common levels would be

80 and 20 respectively. The stochastic indicator is best used in conjunction with support/resistance levels as well as moving averages to broaden the scope of analysis.

3.6.5 Parabolic SAR

The parabolic stop and reverse, SAR, is an indicator used to determine the direction of an asset as well as when to enter or exit the market (Drakopoulou, 2016). The SAR is visualized as a set of dots either above or below the price. The dots also follow the recent trend of the market, with stronger trends leading to steeper slopes in dots. Eventually the dots will meet the price of the asset, causing the dots to flip and quickly move away from the price. Shown below in Figure 4 are SAR dots plotted with example price data.

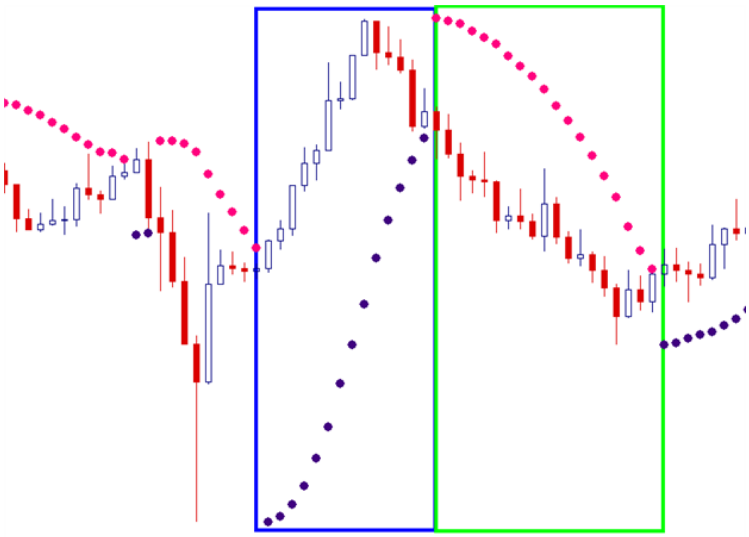


Figure 4: SAR (Parabolic SAR, 2011)

When trading with the SAR, a long trade would be placed when the dots switch from above an asset to below. This would signal that an uptrend has started. When placing a short trade, the dots should switch from below the price of an asset to above. Like all technical

indicators, SAR can be used in a variety of ways. Some examples include using other indicators such as long term averages and SAR for confirmation of an entry, or determining stop losses based on the current SAR dot. It is also important to note that SAR is not very valuable in a sideways market because the dots will frequently flip with no clear price direction.

3.6.6 Average True Range

The average true range (ATR) is a technical analysis indicator that measures market volatility by decomposing the entire range of an asset price for a specific period (Cohen, 2016). Normally, it is derived from a 14 day moving average of a set of true range indicators. In order to calculate average true range, equation 2 is used. Figure 5 shows an example of the ATR in the bottom portion of the price graph.

$$TR = \max[(high - low), abs(high - close_{previous}), abs(low - close_{previous})] \quad \text{(Eq. 2)}$$

Figure 5: Average True Range indicator in Metatrader

Figure 5 shows how the average true range indicator is shown in the Metatrader platform. It mainly ranges from 0.002 to 0.0001. As the value of the average true range is directly proportional to the volatility of the market, the traders can use this indicator for detecting when to go in and out of trades based on the value of the indicator.

3.7.7 Bollinger Bands

Bollinger Bands are used to indicate whether the price of an asset is relatively high or low, and do not give an absolute buy or sell signal (Bollinger, 1992). The bands themselves are plotted two standard deviations above and below a moving average. The bands form an envelope around the current price based on previous movement. Shown below in Figure 6 is a bollinger band envelope with the bands, and moving average labeled.

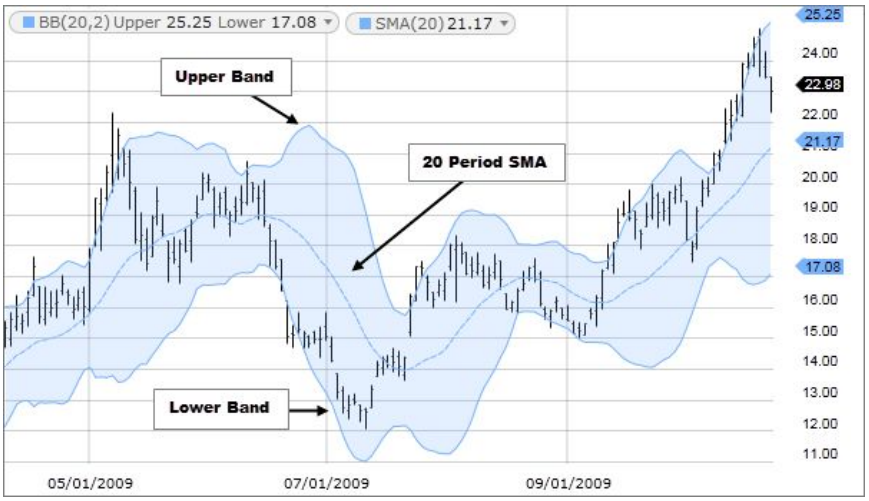


Figure 6: Bollinger Bands (What are Bollinger Bands, 2019)

When trading with Bollinger Bands, it is important to focus on the points when the price approaches either of the bands. When the price is relatively high it will approach the upper band, and when the price is low it will approach the lower band. Bollinger Bands are best used as a confirmation combined with other indicators. For example, if other indicators are signaling sell

and the Bollinger Bands signal that the price is relatively high, it would be a good decision to sell.

These are just a few of the many technical indicators that can help predict trends, momentum, overbought, and oversold conditions of an asset.

4. Optimizing and Analyzing Trading Systems

This Chapter explains the steps needed to optimize and analyze automatic trading systems including backtesting, input optimization, risk management, profitability metrics, and program monitoring. These categories are explored in the context of Metatrader, as many of these topics can vary slightly based on the chosen trading platform.

4.1 Backtesting With Metatrader 5

Backtesting allows a trader to test their developed system on real historical data. One of the greatest benefits of using Metatrader is the ability to backtest easily in their Strategy Tester . The backtesting is performed in the Metatrader Strategy Tester interface. In Strategy Tester, the user can specify multiple inputs that can determine what data is used to test the automatic trading program, referred to in Metatrader as an expert advisor. Figure 7 is the user interface for the Strategy Tester.

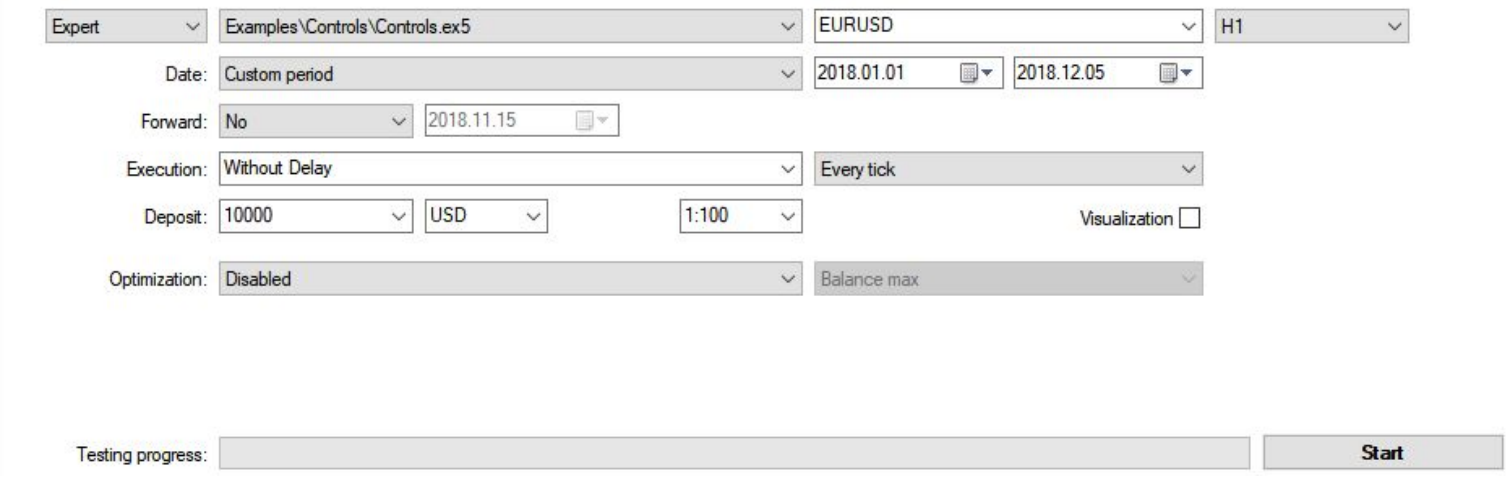


Figure 7: Metatrader Strategy Tester

As seen above, the currency pair, trading time frame, historical time period, initial deposit, etc. can all be varied when backtesting. Once the variables have been determined, the user can start the backtest to get the results of the program over the desired time period. Depending on the time period being tested, and complexity of the program, the Strategy Tester can use all available computational resources (Strategy Testing, 2019). As a result of being CPU dependent, running a test can take a significant amount of time and even slow a computer down.

After the test has completed, a new tab is created which stores the result of the backtest. Figure 8 shows the output of backtesting on an example Metatrader program using the same parameters defined in Figure 7.

History Quality	99%				
Bars	5781	Ticks	25069368	Symbols	1
Initial Deposit	10 000.00				
Total Net Profit	-613.88	Balance Drawdown Absolute	717.31	Equity Drawdown Absolute	815.91
Gross Profit	1 391.39	Balance Drawdown Maximal	717.31 (7.17%)	Equity Drawdown Maximal	821.41 (8.21%)
Gross Loss	-2 005.27	Balance Drawdown Relative	7.17% (717.31)	Equity Drawdown Relative	8.21% (821.41)
Profit Factor	0.69	Expected Payoff	-8.08	Margin Level	7546.03%
Recovery Factor	-0.75	Sharpe Ratio	-0.12	Z-Score	-0.46 (35.45%)
AHPR	0.9992 (-0.08%)	LR Correlation	-0.59	OnTester result	0
GHPR	0.9992 (-0.08%)	LR Standard Error	125.88		
Total Trades	76	Short Trades (won %)	41 (60.98%)	Long Trades (won %)	35 (65.71%)
Total Deals	152	Profit Trades (% of total)	48 (63.16%)	Loss Trades (% of total)	28 (36.84%)
	Largest	profit trade	51.00	loss trade	-218.70
	Average	profit trade	28.99	loss trade	-71.62

Figure 8: An example of Metatrader Backtest

The outputs of the backtest are very detailed and include total profit, percentage of trades won, total trades, as well as other profitability and risk metrics which help to determine the value of the automated trading program.

4.2 Optimization Criteria

The backtesting tool is very convenient, but a problem occurs when trying to test a program multiple times with different input criteria. Input criteria could be any numeric variable that can be changed in the program that would result in a different output. Some examples include amount of lots traded, or variables used in coding the technical indicators. Metatrader allows the user to add input criteria and perform optimization testing. Figure 9 shows the inputs of an example Metatrader program.

Variable	Value	Start	Step	Stop
☐ Lots	0.1	0.1	0.01	1.0
☐ Take Profit (in pips)	50	50	1	500
☐ Trailing Stop Level (in pips)	30	30	1	300
☐ MACD open level (in pips)	3	3	1	30
☐ MACD close level (in pips)	2	2	1	20
☐ MA trend period	26	26	1	260

Figure 9: Metatrader Inputs

The inputs have an initial value that the program will use if run only once. The step value is how much each input is increased in each run of the test. If optimization is enabled in Metatrader Strategy Tester, the application will run all different combinations of the inputs beginning from the Start value, increment by the Step value, and end at the Stop value. In Figure 9, Lots would have 91 steps, Take Profit would have 451 steps, etc. Metatrader would have to run 91 x 451 x 271 x 28 x 19 x 235, or 1.39×10^{12} iterations of the program. This is a prohibitive number of tests when running on a personal computer because of the time needed to run each test.

Metatraders solution was to add another option in Strategy Tester which implements a genetic based algorithm, rather than testing every option. The genetic algorithm picks two random populations, backtests those populations individually, then performs some crosses between populations to determine optimal results (Optimization Types, 2019). Using the algorithm drastically reduces the time needed to run all desired steps. The results of optimization are not as detailed as one backtest, but they show the number of trades, profit, and input criterion used in that test run. This data can then be used to select the best inputs for the automatic trading program.

4.3 Monte Carlo Analysis

Monte Carlo simulation is used mainly for probability simulation that measures impact of risk and uncertainty variables in terms of forecasting or financial situations. The way to employ the simulation is to model the possible movements on currency prices. There are two components to price movements: drift and random input. Drift is a constant directional movement and the random input would represent market volatility. To determine the drift, standard deviation, variance, and average price movement, we would need to analyze historical price data. Luckily, there are websites that use randomized historical data to account for market variations as well as randomly skip position entry to account for execution problems. Figure 10 shows the original data for EURUSD prices starting from 2017-08-03 01:55 to 2017-11-08 13:35. Figure 11 shows the monte carlo simulation of the very data.

To project one possible priceline, we need to use historical price data to generate a series of daily returns. The equations used in Monte Carlo simulation are shown below.

$$periodic\ daily\ return = ln(day's\ price / previous\ day's\ price) \quad \text{(Eq. 3)}$$

$$drift = average\ daily\ return - (variance/2) \quad \text{(Eq. 4)}$$

$$random\ value = standard\ deviation * NORMSINV(RAND()) \quad \text{(Eq. 5)}$$

$$next\ day's\ price = today's\ price * e^{(drift+random\ value)} \quad \text{(Eq. 6)}$$

Each equation gives the set value for each of the variables that are necessary for each separate line. Different random values result in different price trajectories. As you can see in Figure 11, each different random value results in a different colored line.

Figure 10: Original data of EURUSD chart

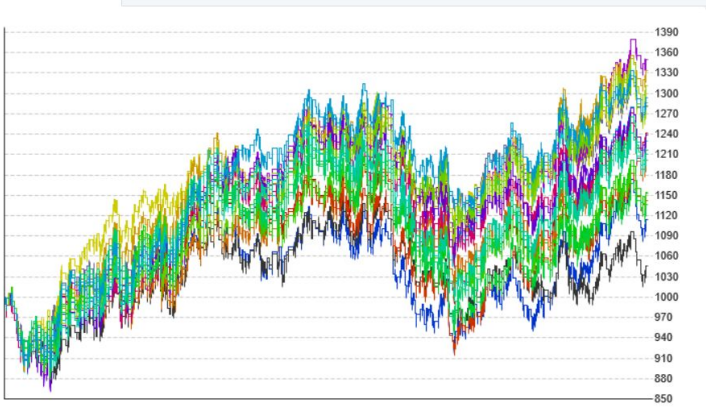


Figure 11: Monte Carlo simulation on EURUSD chart

4.4 Risk

One of the most important factors in financial trading is risk management. In order for the traders to end up on top of the trades, the trader needs to manage his risks because even a few oversized losses can outweigh the winnings of many other trades. For handling such situations, the trader should add the stop-loss and take profit conditions in their programs to reduce maximum loss, and prevent additional loss while they are on the winning side of the trade. In order to handle how much return the trader receives for the extra volatility you endure for holding a riskier asset, a measure called Sharpe's ratio is formed. To calculate Sharpe's ratio use equation 7 shown below.

$$TS(x) = (r_x - R_f) / StdDev(x) \quad \text{(Eq. 7)}$$

In equation 7, x is the investment, r_x is the average rate of return of x, R_f is the best available rate of return of a risk-free security, and StdDev(x) is the standard deviation of r_x.

4.5 Profitability Metrics

Determining whether a system does well or not depends on several factors. One obvious factor would be "does it make money?" However, this is not as simple as it may seem. A system may make a profit over one year, but over several years it may lose money. Typically portfolios are tracked over a 10-year period. This time frame tends to level out sudden drops or spikes in price and usually gives a balanced market trend. If a system is consistently profitable over 10 years, then the system can be considered successful.

Not all systems that break even over 10 years are considered profitable. Generally a system needs to perform at a certain level to be considered a contender in the financial industry. A good way of measuring the profitability of a system is to compare it against indexes. The most popular comparison index is the S&P 500. If a system can outperform the S&P 500, the system is profitable. Combining this with the 10-year period, if a system can outperform the S&P 500 over a 10-year period, the system is trending towards being financially stable and profitable, which is what investors look for in a fund.

4.6 System Monitoring

Because of the nature of a completely autonomous system, mistakes are bound to occur. As such, it is imperative that the user regularly monitors the program to ensure nothing fails. There are measures in place that aim to prevent loss, such as stop-losses, but these are not perfect and sudden price changes may interfere with them. One such problem may result from a news article or press release, which can cause prices to rise or fall dramatically in a short period of time. If a user sees such an event, it would be wise to close all trades in progress and wait until the market levels out.

5. Results

This Chapter describes the functionality and resulting predictions of each of the three individual systems created including the program methodology, inputs, optimization, and backtesting process. The results of each individual system are presented and analyzed to determine the systems quality.

5.1 Individual System 1 (Jake Barefoot)

5.1.1 Methodology

The goal of this system was to use a plus/minus combination of different indicators that look at momentum, trend, and overbought/oversold indicators to determine positioning. The indicators used include Parabolic SAR, Triple Exponential Moving Average(TRIX), MACD, RSI, and Bollinger Bands. The TRIX indicator is an exponential moving average of an exponential moving average of an exponential moving average. This would give a smoothed and recently weighted version of a standard moving average. Each of these indicators tell one particular condition of the market such as momentum or trend. The combination of these indicators can help determine the optimum time to trade in a market. If there are multiple buy signals from all different types of indicators it should in theory be an opportune time to buy.

This system was designed and tested in Metatrader 5 with the following parameters. The program would be trading with EURUSD. Because EURUSD is such a frequently traded currency pair, the price swings are less drastic and less volatile than other pairs. Less movement does mean less potential for profit, but the price patterns are more predictable and stable.

Another parameter used was the 30 minute trading time frame. The employed indicators are calculated using this time frame because the trading system was designed to get in and out of the market quickly. A faster time frame compared to the daily and weekly time frames, shows more accurately the current state of the market. The last parameter set for this program was the deposit volume of $1,000,000. The specific amount is not extremely important because the program will be evaluated based on percentages rather than totals. The main advantage of keeping the initial amount in a multiple of $100,000 is the convenience of buying multiple lots in Metatrader.

5.1.2 Program Flow

The program was created using the MQL5 coding language on the MetaEditor platform and designed to run with the parameters specified above. Shown in Figure 12 is the program flow diagram of the system.

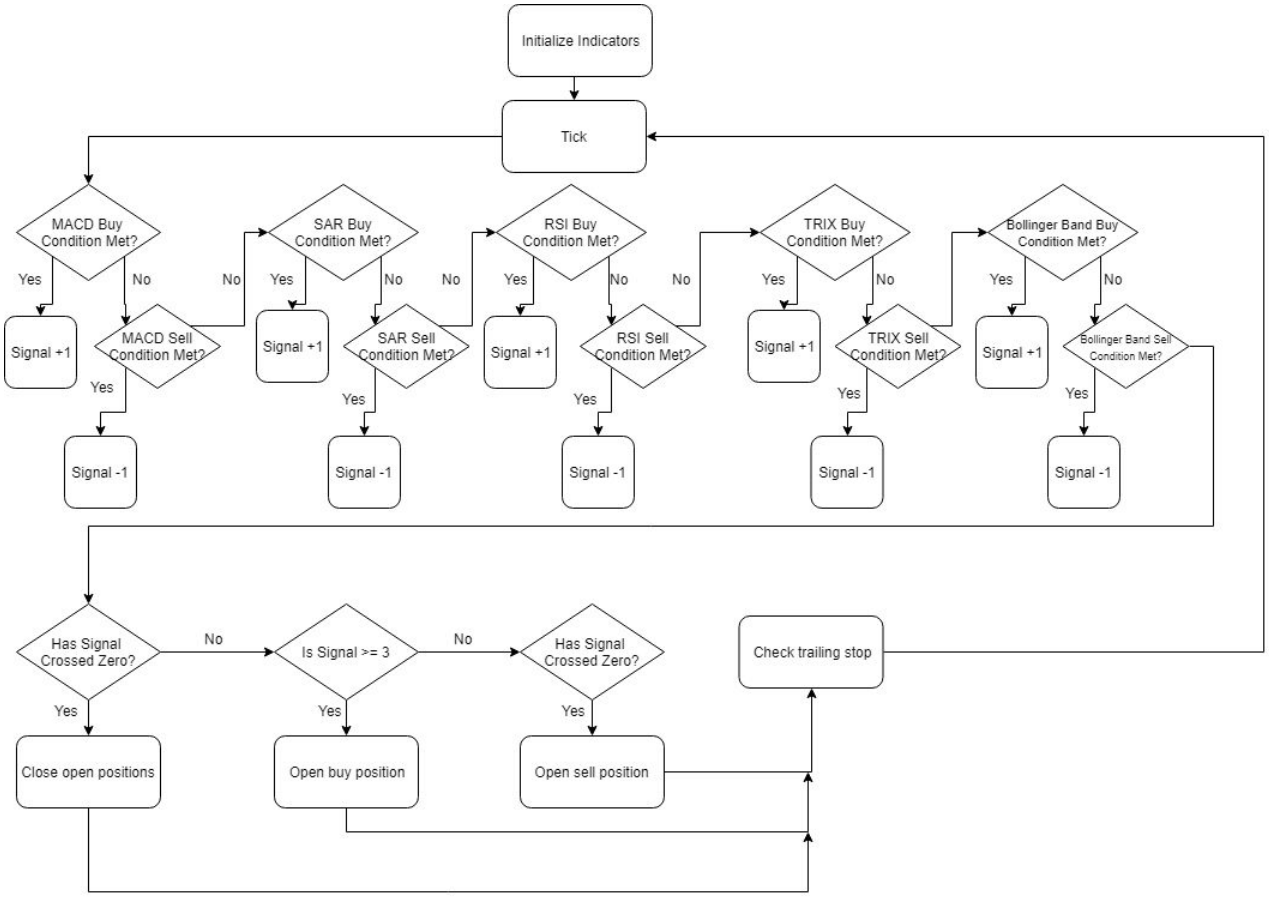


Figure 12: Program Flow Diagram of System 1

The program runs in a main loop that is controlled by the currency market. The tick block represents the start of the loop that initiates every time there is a registered price change of EURUSD. Once there is a detected change in price, the program will then check all of the indicators to determine if the buy or sell condition has been met. If the buy condition is met, then the signal variable will be incremented. If the sell condition is met, then the signal variable will be decremented. After all indicators have been checked, the program observes the signal variable to determine whether to buy, sell, or close a trade. For either a buy or sell trade, all previous trades are closed. It is also important to know that the position size of a trade depends on the buy

signal. When a trade is executed, the program will create an order with the amount of lots as the signal value. For example if the signal is -4, the program will close previous orders and open a new sell trade with a position size of 4 lots. After the trading decisions have been made, the trailing stop loss value will be checked. If there is still an active trade, the stop loss for that trade is augmented if necessary. The full system code for this program can be found in Appendix A.

5.1.2 Inputs

As seen in Appendix A there are 15 inputs for this program. The inputs represent different indicator values and trade parameters. For each indicator, there are a certain numeric values that change the indicator output. For example, The TRIXperiod input is the number of previous bars used for the triple exponential moving average indicator. This value is set to be 18 in the final program meaning that the moving average will look at the 18 previous 30 minute price points to create the indicator. Changing any of these indicators can drastically alter how the program makes a trade, so it is extremely important to find the values that produce the best results. In order to do this, optimization needed to be performed on the program to find the ideal input values.

5.1.3 Optimization

As discussed in Chapter 4, optimization is convenient in Metatrader because of its access to historical data. The first step in the optimization process is determine the range of values we want to look at for each input. Shown in Figure 13 are the different values used for optimization of this program.

Variable	Value	Start	Step	Stop	Steps
SARstep	0.03	.01	.01	.03	3
SARmax	.3	.2	0.1	.3	2
TRIXperiod	18	12	2	18	4
MACDfast	10	10	1	14	5
MACDslow	25	25	1	28	4
MACDsignal	10	7	1	11	5
RSIperiod	8	8	1	12	5
RSIupper	60	60	5	75	4
RSIlower	40	25	5	40	4
BANDperiod	14	14	2	22	5
BANDsd	2	1.4	0.2	2	4
TakeP	400	100	50	500	9
StopI	100	100	50	500	9
SLSmover	350	100	50	500	9
SLBmover	250	100	50	500	9
					25194240000

Figure 13: Optimization Input of System 1

The start, step, and stop values are shown for the optimization process. Even though there are only a few steps for each variable, the total number of combinations for these inputs is over 25 billion. This would take months to run on a desktop computer if testing every combination. To quicken the process we will use the genetic algorithm offered by Metatrader, as well as the 1 minute Open, High, Low, Close (OHLC) price option. Using 1 minute OHLC quickens the testing process by looking at price data over one minute bars, rather than individual ticks. The next step was to determine what data the program would be tested on. Shown in Figure 14 are the settings used for the optimization of this system.

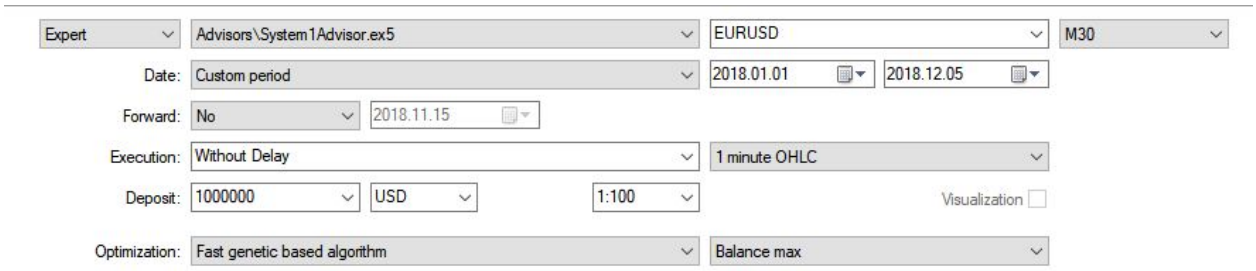


Figure 14: System 1 Optimization Settings

These settings will be consistent throughout the optimization, except for the dates fields. The decision was to test on the first half of the current decade for a total of 4 years from

2010-2013. Each year was tested from January 1 to December 5. This allows for downtime that can be attributed to code maintenance, or manual exit from the market based on upcoming news events. After performing optimization there were 4 tables ordered by profit with approximately 5,000 simulations performed each year.

In order to determine the best value for each input, the average profit for all simulations across years was determined for each step value of all inputs. The step value for each input with the highest profit was chosen for the final program. Table 4 below shows the final input values chosen and their average return value based on over 20,000 simulations performed.

Table 4: Best Input Values for System 1

Input	**Choice**
SARstep	0.03
SARmax	0.3
TRIXperiod	18
MACDfast	11
MACDslow	25
MACDsignal	10
RSIperiod	8
RSIupper	60
RSIlower	40
BANDperiod	16
BANDsd	2
Takep	400
Stopl	100
SLSmover	350
SLBmover	250

After finding the best values for the inputs, the program was complete and ready to be tested on other historical data.

5.1.4 Results

After performing optimization, the results of the program needed to be determined using a new set of data. The time frame chosen for backtesting was 2014-2018. These years were chosen because of the high data history quality in Metatrader. Some older years had less accurate price data. The percentage return for each year is summarized in Table 5.

Table 5: System 1 Return

Year	**Program Return**
2014	11.24%
2015	20.15%
2016	11.40%
2017	8.10%
2018	21.39%
Average	14.46%

The average return for this system over the five year testing period was 14.46%. There are a few key takeaways based on the return values. One takeaway is that there were consistently positive returns with the lowest yearly return being 8.10% and the highest return being 21.39%. This level of return is great for an automated system.

There are other indicators that can signal the quality of a system besides just total profit. Shown in Table 6 are a few of the other testing outputs from the backtests performed.

Table 6: System 1 Measurements

Year	Sharpe Ratio	Total Trades	Average Profit	Average Loss	Profit Trade %
2014	0.13	10209	$ 67.56	$ 42.07	48.42%
2015	0.13	10800	$ 101.19	$ 70.56	51.94%
2016	0.1	10561	$ 72.09	$ 60.25	53.68%
2017	0.09	10593	$ 61.84	$ 48.31	51.26%
2018	0.21	10917	$ 77.99	48.45	53.82%
Average	0.132	10616	$ 76.13	$ 53.93	51.82%

This Table highlights how a system that trades extremely quickly can impact profitability and risk metrics. The system has shown to be very profitable returning nearly 14.5% a year, but the average sharpe ratio is only .132. This would signify that the system is extremely risky, but that is somewhat misleading. There were an average of 10,616 trades made each year, meaning each trade is only active on average for less than 25 minutes. In each of those trades $300,000-$500,000 is being risked. In reality, a currency pair will never drop to $0, or increase infinitely, unless a government completely fails, and EUR/USD will never have extremely volatile swings in such a short time. This means we are not actually risking the total trade value. Additionally, with a 100 pip trailing stop the most a trade can possibly lose is $5,000 or .5% of the initial deposit. Even though some of these metrics are not ideal for a trading system, there were steps taken to mitigate the risk, such as the trailing stop.

One area of concern is the percentage of trades that were profitable. Over all the years tested only 51.82% of trades earned a profit. The total system was so profitable because the average profit trade earned 34.1% more than the average loss trade value. Maintaining more consistent profitability per trade would be a key area of focus when implementing future changes to the system. The full testing data for this system can be found in Appendix D.

5.2 Individual System 2 (Michael Cullen)

5.2.1 Methodology

Having no coding experience, for me this project was mostly learning how to trade forex, learning the MQL5 language, and experimenting with different systems I created. Therefore my methodology for this project were based on trial and error.

Once I had a basic understanding of foreign exchange trading, I looked into a broad selection of indicators to help give me an idea of what I was working with and what would be beyond my reach.

The main problem I encountered was learning the MQL5 language. Since it is a relatively new programming language, and Metatrader is the only platform that uses it, there are very few tutorials. Luckily, R. Bauer (Bauer) has an extensive video tutorial series covering basics of forex trading, the Metatrader system, and hundreds of videos explaining how to create various indicators within MetaEditor, Metatrader's code editing program. From these, I created roughly two dozen codes containing variations of these indicators, each of these having gone through dozens of edits to create a working code with varying success.

5.2.2 Program Flow

My program flow was not a typical program flow. Specifically, the process of deciding which system to utilize was based on trial and error of many different types of programs, which are outlined in Figure 15.

Conditions	Timeframe	Period	Exec.	Results	Parameters	Comments
RSI>70 sell, RSI<30 buy, SL 50 points	1 year	1 H	Every tick	-60%		
RSI>70 sell, RSI<30 buy, SL 100 points	1 year	1 H	Every tick	-60%		
RSI<30 buy, TSL 50 points, +10	1 year	1 H	Every tick	-80%		
RSI<30 and ask<lowerbollinger buy	1 year	1 H	Every tick	Failure		
Long SAR, SL 50, TP 200	1 year	1 H	Every tick	-7%		
Long and Short SAR, SL 50, TP 200	1 year	1 H	Every tick	5%		
Long and Short SAR, closepositions on switch	1 year	1 H	Every tick	0%	Failure, would not close positions	
SAR, close positions if (lastvalue-nexttolastvalue)>50	1 year	1 H	Every tick	0%	Buys at correct time, but does not close	
SAR, TP 100-300 by 20,	1 month	1 H	Every tick	0.10%	TP 250	
SAR, TP 250-350 by 10, SL 100-200 by 10	1 month	1 H	Every tick	0.59%	TP 300, SL 130	
SAR, TP 290-310 by 1, SL 120-140 by 1	1 month	1 H	Every tick	0.63%	TP 307, SL 130	
SAR, TP 307 by 1, SL 130	1 year	1 H	Every tick	0.20%		Simply optimized that month, not in general
RVI, TP 100-300 by 20, SL 50-150 by 20	1 year	1 H	Every tick	3.90%	TP 280, SL 150	
RVI, TP 270-300 by 4, SL 140-180 by 5	1 year	1 H	Every tick	6.40%	TP 278, SL 140	
SAR+RVI, TP 250-300 by 10, SL 100-150 by 10, ADX 25-60 by 5	1 year	1 H	Every tick	0.74%	TP 260, SL 110, ADX 25	
SAR+RVI, TP 250-270 by 2, SL 100-120 by 1, ADX 15-30 by 2	1 year	1 H	Every tick	3.47%	TP 268, SL 100, ADX 21	
SAR+RVI, TP 260-270 by 1, SL 80-110 by 2, ADX 17-23 by 1	1 year	1 H	Every tick	4.04%	TP 266, SL 98, ADX 21	
SAR+RVI, TP 264-268 by 1, SL 95-100 by 1, ADX 21	1 year	1 H	Every tick	4.35%	TP 265, SL 98, ADX 21	Only 31.45% win, close position needs work
SMA<price buy, SMA>price sell, SL 50, timeframe 5-50 by 2	1 year	1 H	Every tick	6.70%	7 frames	
SMA<price buy, SMA>price sell, SL 50-150 by 10, timeframe 5-9 b	1 year	1 H	Every tick	6.84%	SL 50, 8 frames	
SMA<price buy, SMA>price sell, SL 25-60 by 5, timeframe 5-9 by	1 year	1 H	Every tick	6.84%	SL 50, 8 frames	
SAR buy, SMA>price sell	1 year	1 H	Every tick	Failure		
SAR, SL 50-150 by 10	1 year	1 H	Every tick	3%	SL 50	
SAR, SL35-55 by 1	1 year	15 M	Every tick	3.80%	SL 41	Not selling on switch
Long SAR+TSL	1 year	1 H	1 M OHLC	7%		Visualization didn't match profit
Long SAR+optimized TSL	1 year	15 M	1 M OHLC	No trades		
Long SAR+optimized TSL	1 year	1 H	1 M OHLC	34%*	SLValue 80, SLModifier 10	I thought I had figured something out
Long SAR+optimized TSL	1 year	1 H	1 M OHLC	61%*	SLValue 25, SLModifier 10	Knew something was wrong but optimistic
Long SAR+optimized TSL	1 year	15 M	1 M OHLC	64%*	SLValue 15, SLModifier 10	Definitely knew something was wrong
Long SAR+optimized TSL	1 year	15 M	Real ticks	-90%		1M OHLC did not represent it well
Long SAR+optimized TSL	1 year	30 M	Real ticks	>1%		Realized optimization might not be working
Long SAR+TSL	1 year	1 H	Real ticks	-4%	SLValue 150, SLModifier 100, Adder 10	
Long SAR+TSL+RSI 25-45 by 2	1 year	1 H	Real ticks	0%	SLValue 150, SLModifier 100, Adder 10	
Long SAR+TSL+RSI 25-45 by 2	1 year	1 H	Real ticks	0%	SLValue 200, SLModifier 100, Adder 10	
Long SAR+TSL+RSI 25-45 by 2	1 year	1 H	Real ticks	0%	SLValue 1000, SLModifier 100, Adder 10	Optimization still broken
Long SAR+TSL+RSI<35	1 year	1 H	Real ticks	0%	SLValue 150, SLModifier 100, Adder 10	16/46 trades won
Long SAR+TSL+RSI<35	1 year	1 H	Real ticks	-1%	SLValue 250, SL Modifier 100, Adder 10	15/47 trades won
Long SAR+TSL+RSI<35	1 year	1 H	Real ticks	-2%	SLValue 500, SL Modifier 400, Adder 10	3/12 trades won
Long SAR+TSL+Envelopes	1 year	1 H	Every tick	0%	SLValue 500, SL Modifier 400, Adder 10	Only 4 trades at marginal levels
Long SAR+TSL+Envelope-3Shift	1 year	1 H	Every tick	0%	SLValue 500, SL Modifier 400, Adder 10	Buying and selling at odd times, adjusting code syntax
Long SAR+TSL+Envelope-3Shift	1 year	1 H	Every tick	0%	SLValue 500, SL Modifier 400, Adder 10	Same results, code wasn't the problem Adjusting SAR step to 0.005 to reduce noise
Long SAR+TSL+Envelope-3Shift	1 year	1 H	Every tick	0%	SLValue 500, SL Modifier 400, Adder 10	No trades, removing envelope
Long SAR+TSL	1 year	1 H	Every tick	-99%	SLValue 500, SL Modifier 400, Adder 10	Buying, immediately selling, then buying again, losing thousands per trade, removing SL and going to normal SAR
SAR	1 year	1 H	Every tick	11.56%		Only one trade, which won. Need to fix closeposition
Long SAR	1 year	1 H	Every tick	-1%		Sell simply sold another instead of closing position
Long SAR	1 year	1 H	Every tick	-2%		Fixed positionclose, however I think timeframe is off
Long SAR	1 year	15 M	Every tick	-9%		Need stop loss
Long SAR, SL 50-500 by 10	1 year	1 H	Every tick	-0.50%	SL 490	Better, still not great
Long SAR, SL 200-500 by 10	1 year	1 H	Every tick	-0.16%	SL 200	Step: 0.02, Max 0.05
Long SAR, SL 100-300 by 10	1 year	1 H	Every tick	0.29%	SL 170	Getting there, still need better methods
Long SAR+TSL	1 year	1 H	Every tick	-0.35%	SLValue 500, SL Modifier 400, Adder 10	Step: 0.02, Max 0.05
Long SAR+TSL	1 year	1 H	Every tick	-0.45%	SLValue 300, SL Modifier 200, Adder 10	
Long SAR, SL 100	1 year	1 H	Every tick	-2.30%		
Long SAR, SL 100	1 year	30 M	Every tick	-3.60%		
Long SAR, SL 100	1 year	15 M	Every tick	-11%		
Long SAR, SL 100	1 year	2 H	Every tick	-4.60%		
Long SAR, SL 200	1 year	1 H	Every tick	-1.60%		
Long SAR, SL 200	1 year	30 M	Every tick	-6.50%		
Long SAR, SL 200	1 year	15 M	Every tick	-7.12%		
Long SAR, SL 200	1 year	2 H	Every tick	-5.85%		
Double SMA	1 year	1 H	Every tick	2.09%	Short=16, Long=58, SL=100, TP=170	
Double SMA	1 year	1 H	Every tick	3.53%	Short=12, Long=70, SL=150, TP=140	
Double SMA	1 year	1 H	Every tick	3.76%	Short=11, Long=74, SL=150, TP=140	
Double SMA	1 year	1 H	Every tick	3.76%	151 trades, 60% won	
Double SMA	10 year	1 H	Every tick		History corrupt	

Figure 15: Forex Test Log

During the trial period, 65 test were recorded. Many additional tests were conducted, but many did not run or did not complete any trades and these were omitted. I started experimenting with RSI, but they consistently proved to lose drastic amounts of money. I then tried SAR with a stop loss and take profit, which needed refining. Initial testing yielded some success, but mostly due to a well-timed backtesting timeframe. Other tests broke even. I then tried a RVI, followed by a RVI combined with a SAR which seemed promising. I added a ADX, which gave results of approximately 4%, but the win/loss ratio was only 31%. I abandoned the idea for a while and started at the beginning with a basic simple moving average code, which gave a profit of up to 6.8%. However, this again was due to the timeframe and did not work on other years.

After these initial tests, I decided that SAR seemed the most promising and focused heavily on getting the code to work. Trading SAR on the long and short did not seem to work well as they cancelled each other out. I had to focus on only long trades. I added a trailing stop loss parameter, which did not do much. I then supplemented the code with a RSI because it seemed the code was trading even when the market was ranging. Unfortunately the RSI effect was too great, because it did not trade enough. I tried replacing the RSI with envelopes, but it did not trade at all.

At this point I re-wrote the code because it seemed the program itself was not trading enough. However, since I discovered this was not the case, I then tried varying the timeframes, the period, the ticks, and different stop loss values without success. I decided to abandon the SAR idea and return to the SMA approach, which had performed decently. I decided to use a double SMA to help mitigate false signals and had relatively consistent results of roughly 3%. However, when backtested over 10 years, there was a slight loss overall. I then realized the code had an error, which required a retest.

5.2.2 Inputs

The code itself is relatively simple. First, the program creates arrays for the short and long moving averages and creates a series for each of the arrays. Then it creates a buffer from the array and series to determine which values it uses. The main protocol in the code is determining when to buy or sell the code. It will buy when the short moving average passes above the long moving average and sell when the short moving average passes below the long moving average. The full code for my system can be found in Appendix B.

The adjusted inputs in the system code are “SMATimeframe” for the short moving average timeframe, “LMATimeframe” for the long moving average timeframe, “StopLossValue” for the stop loss value, and “TakeProfitValue” for the take profit value.

5.2.3 Optimization

I started the optimization process very broad and slowly refined it. The first step was to test each parameter at extreme values and slowly bracket the value in. Standard short moving average timeframes are between 3 and 20 intervals, long moving average timeframes are between 30 and 80, and I took guesses at stop loss and take profit values.

Variable	Value	Start	Step	Stop	Steps
☑ SMATimeframe	11	5	2	15	6
☑ LMATimeframe	74	45	5	75	7
☑ StopLossValue	150	100	20	300	11
☑ TakeProfitValue	140	100	20	300	11
					5082

Figure 16: First Optimization Step

After several optimization runs, Figure 17 represents the final optimization values. The best values were as follows: SMA of 14, LMA of 37, SL of 250, and TP of 280. This yielded 118% from Jan 1, 2017 to Jan 1, 2018.

Variable	Value	Start	Step	Stop	Steps
☑ SMATimeframe	14	12	1	14	3
☑ LMATimeframe	37	35	1	40	6
☑ StopLossValue	250	230	10	310	9
☑ TakeProfitValue	280	170	10	320	16
					2592

Figure 17: Final Optimization Step

5.2.4 Results

This code has the potential to be very profitable. From Jan 1, 2016 to Jan 1, 2017, the initial deposit was more than doubled. Figure 18 shows the balance and equity of the system from January 2016 to January 2017.

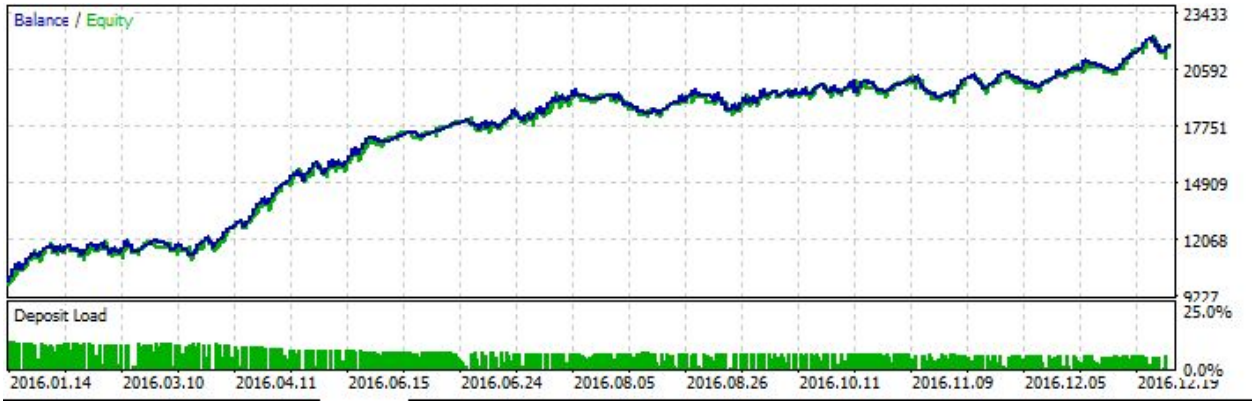


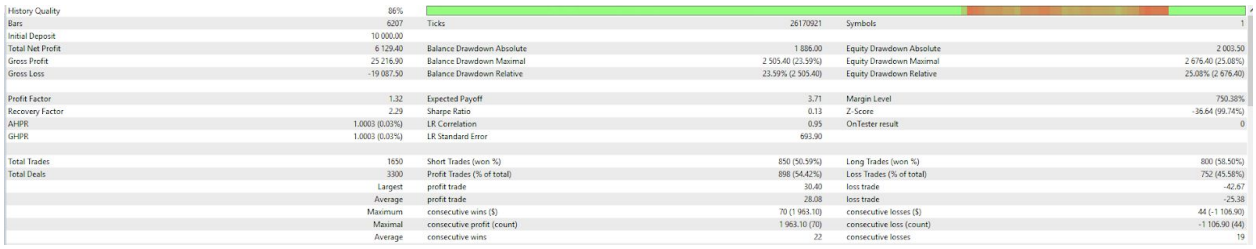
Figure 18: 2016-17 Balance/Equity Graph

Based on a total profit of $11,802 with a 1:100 leverage and 2046 trades, 56.17% of the 1127 short trades won and 61.15% of the 919 long trades won. As shown in Figure 19, 1195 trades won and 851 trades lost.

History Quality	99%				
Bars	6227	Ticks	35237378	Symbols	1
Initial Deposit	10 000.00				
Total Net Profit	11 802.80	Balance Drawdown Absolute	0.00	Equity Drawdown Absolute	144.00
Gross Profit	33 562.97	Balance Drawdown Maximal	1 199.10 (6.12%)	Equity Drawdown Maximal	1 301.50 (6.62%)
Gross Loss	-21 760.17	Balance Drawdown Relative	6.12% (1 199.10)	Equity Drawdown Relative	8.13% (982.00)
Profit Factor	1.54	Expected Payoff	5.77	Margin Level	913.64%
Recovery Factor	9.07	Sharpe Ratio	0.23	Z-Score	-40.03 (99.74%)
AHPR	1.0004 (0.04%)	LR Correlation	0.94	OnTester result	0
GHPR	1.0004 (0.04%)	LR Standard Error	1 125.25		
Total Trades	2046	Short Trades (won %)	1127 (56.17%)	Long Trades (won %)	919 (61.15%)
Total Deals	4092	Profit Trades (% of total)	1195 (58.41%)	Loss Trades (% of total)	851 (41.59%)
	Largest	profit trade	29.50	loss trade	-59.57
	Average	profit trade	28.09	loss trade	-25.57
	Maximum	consecutive wins ($)	61 (1 711.10)	consecutive losses ($)	40 (-1 015.10)
	Maximal	consecutive profit (count)	1 711.10 (61)	consecutive loss (count)	-1 015.10 (40)
	Average	consecutive wins	21	consecutive losses	15

Figure 19: 2016-17 Backtesting Results

2017-2018 was very similar, except the history quality was relatively low. The system still made 61.29%, but that may be due to history errors. Figure 20 shows the history error in the red block in the green history bar.



History Quality	86%				
Bars	6207	Ticks	26170921	Symbols	1
Initial Deposit	10 000.00				
Total Net Profit	6 129.40	Balance Drawdown Absolute	1 886.00	Equity Drawdown Absolute	2 003.50
Gross Profit	25 216.90	Balance Drawdown Maximal	2 505.40 (23.59%)	Equity Drawdown Maximal	2 676.40 (25.08%)
Gross Loss	-19 087.50	Balance Drawdown Relative	23.59% (2 505.40)	Equity Drawdown Relative	25.08% (2 676.40)
Profit Factor	1.32	Expected Payoff	3.71	Margin Level	750.38%
Recovery Factor	2.29	Sharpe Ratio	0.13	Z-Score	-36.64 (99.74%)
AHPR	1.0003 (0.03%)	LR Correlation	0.95	OnTester result	0
GHPR	1.0003 (0.03%)	LR Standard Error	693.90		
Total Trades	1650	Short Trades (won %)	850 (50.59%)	Long Trades (won %)	800 (58.50%)
Total Deals	3300	Profit Trades (% of total)	898 (54.42%)	Loss Trades (% of total)	752 (45.58%)
	Largest	profit trade	30.40	loss trade	-42.67
	Average	profit trade	28.08	loss trade	-25.38
	Maximum	consecutive wins ($)	70 (1 963.10)	consecutive losses ($)	44 (-1 106.90)
	Maximal	consecutive profit (count)	1 963.10 (70)	consecutive loss (count)	-1 106.90 (44)
	Average	consecutive wins	22	consecutive losses	19

Figure 20: 2017-18 Backtesting Results

However, the code is also very risky. During certain years the system would generate a return of over 100%, while during other years it could lose up to 90%. Specifically, 2015-16 was a particularly bad year where 99% of the deposit was lost. Figures 21 and 22 show the results of the system from January 2015 to January 2016.

History Quality	99%				
Bars	6195	Ticks	29836358	Symbols	1
Initial Deposit	10 000.00				
Total Net Profit	-9 910.15	Balance Drawdown Absolute	9 910.15	Equity Drawdown Absolute	9 910.15
Gross Profit	21 620.70	Balance Drawdown Maximal	9 910.15 (99.10%)	Equity Drawdown Maximal	9 960.65 (99.11%)
Gross Loss	-31 530.85	Balance Drawdown Relative	99.10% (9 910.15)	Equity Drawdown Relative	99.11% (9 960.65)
Profit Factor	0.69	Expected Payoff	-4.90	Margin Level	63.29%
Recovery Factor	-0.99	Sharpe Ratio	-0.12	Z-Score	-39.76 (99.74%)
AHPR	0.9978 (-0.22%)	LR Correlation	-0.96	OnTester result	0
GHPR	0.9977 (-0.23%)	LR Standard Error	648.94		
Total Trades	2022	Short Trades (won %)	985 (42.13%)	Long Trades (won %)	1037 (33.94%)
Total Deals	4044	Profit Trades (% of total)	767 (37.93%)	Loss Trades (% of total)	1255 (62.07%)
	Largest	profit trade	32.70	loss trade	-26.20
	Average	profit trade	28.19	loss trade	-25.12
	Maximum	consecutive wins ($)	56 (1 591.90)	consecutive losses ($)	70 (-1 757.10)
	Maximal	consecutive profit (count)	1 591.90 (56)	consecutive loss (count)	-1 757.10 (70)
	Average	consecutive wins	14	consecutive losses	22

Figure 21: 2015-16 Backtesting Results

Figure 22: 2015-16 Balance/Equity Graph

Over 10 years, from Jan 1, 2007 to Jan 1, 2017, the system lost essentially the entire account. Figure 23 displays the equity and balance over this timeframe. It is also clear in this chart that as the balance drops, deposit load goes up. This is due to the system trading the same lot size despite there not being enough equity in the account.

Figure 23: 10-Year 2007-17

The poor performance may also be due to history quality, as longer timeframes have poorer history quality, but I feel it may be due to years where huge losses occur. Overall, the system seems to be completely random based on the years themselves rather than the system. A possible way to fix this would be to add a market strength indicator such as RSI or RVI to help prevent bad trades. It may also help to add some position close parameters other than a simple

take profit or stop loss. Also, in order to prevent total loss of the account, a system should only trade a certain percentage of the deposit instead of a fixed lot size.

5.3 Individual System 3 (Aung Min)

5.3.1 Methodology

The main purpose of creating this system is to perfect a program that buys and sells based on the trends of the basic forex indicators solely optimized for the EURUSD currency pair. The indicators involved within this trading bot are Relative Strength Index, Stochastic Indicator, and Average True Range. RSI and ATR values are based on the last 14 periods, Stochastic D-value is based on the last 5 periods, and Stochastic K-value is based on the last 3 periods using 30 minute time-frame. With initial deposit set to $10000, the program will determine whether to buy or sell 0.1 lot depending on the indicators listed above.

5.3.2 Program Flow

The logic behind this trading program is that it would check the volatility of the currency pair using the ATR indicator, as well as checking the overbought and oversold values based on both the RSI and Stochastic indicators simultaneously in an effort to determine a buy or a sell condition. Figure 24 depicts the overall coding approach that was adopted to develop the program described above.

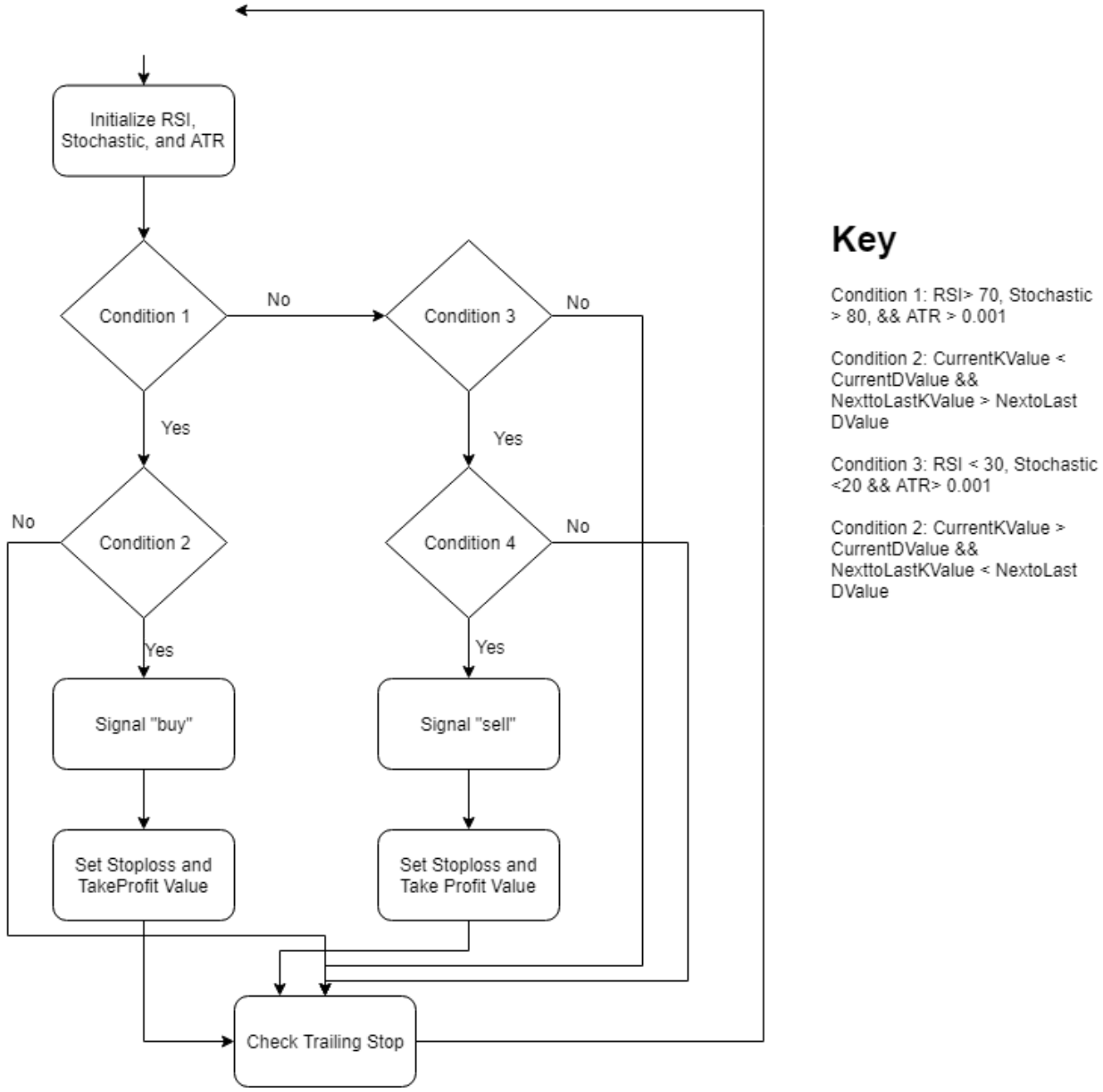


Figure 24: The Flowchart of Aung's Trading Program

The on tick function will run as a loop for this program. Every update to the price of EURUSD on Metatrader 5 will trigger this function to work. First off, the program will initialize the values for the RSI array, ATR array, Stochastic D-value and K-value arrays. After initializing, it will extract the most recent values from the arrays and use them to check the

conditions listed above in Figure 24. The conditions RSI value is greater than 70 and Stochastic value is greater than 80 are used for checking the oversold value of the currency. If they are both signaling oversold, and ATR shows that there is enough market volatility, it is a decent time to enter the market. Although the conditions are proven to be correct, it is always better to check whether the momentum of the price by checking the K-value and D-value shown in condition 2 of the Figure 24. When the momentum shows that the price is going to fall back, it will inevitably sell. The same goes for the selling part of the system. First, the program will check whether the RSI value is less than 30 and the Stochastic value is less than 20. If the conditions are satisfied, the program will check the volatility of the market. If all these conditions passed, it is a clear indication of having an overbought situation in the market. Just like the selling logic, the K-value and D-value of stochastic will be checked as shown in condition 4 of Figure 24. If the momentum is going up, the program will make a buy order of 0.1 lot. A trailing stop-loss is implemented to modify the buy or sell position based on a stop-loss that changes by the current price. The trailing stop-loss() function runs every time there is a tick so that the program checks at every price point whether to increase or decrease the price of the buy or sell order.

5.3.2 Inputs

The inputs for this system would be ten thousand dollars starting amount and 1 minute OHLC data that is fed into the system for it to trade based on. In order to get the values of the indicators, the time-frame that is inputted into the system is thirty minutes apart. The indicators inputted into the system are Stochastic, RSI, and ATR.

5.3.3 Optimization

In trading, every value associated with the indicators has its own purpose. It would be very ideal to be able to optimize every value due to the lack of computing power. The variables in Figure 25 were identified manually due to their importance to the system. SLI and SLImover variables are mainly for the indication of when to have the stop loss to start. SLI is for buy order and SLImover is for sell orders. RSI_Upper and Stoch_Upper are for optimizing the oversold conditions of the RSI and Stochastic indicators for buy orders and RSI_Lower and Stoch_Lower are for optimizing the overbought conditions of the RSI and Stochastic indicators for the sell order. The ATR_Value is for picking the best volatility value for the program to enter the market. The stoplossvalue variable is mainly for optimizing the amount of pips that should be set as stop-loss for both buy and sell orders.

Variable	Value	Start	Step	Stop
SLI	300	150	50	300
RSI_Upper	65	65	5	80
RSI_Lower	35	20	5	35
Stoch_Upper	80	60	5	80
Stoch_Lower	20	20	5	30
ATR_Value	0.001	0.001	0.001	0.003
SLImover	100	100	50	200
stoplossvalue	10	10	5	20

Figure 25: Optimization Inputs of the System

After optimization, the data from each year is then manually entered into pivot tables to calculate the output for the system. Figure 26 shows the result of the optimization done. Each of the calculated by getting the average value from each year then averaging the nine average values across nine years.

RSI	65	70	75	80
20	10615.15	10371.4	10211.28	10154.05
25	10644.24	10544.62	10357.59	10264.46
30	10604.17	10508.67	10446.93	10348.28
35	11095.1	10751.88	10584.27	10432.27

STOC	60	65	70	75	80
20	10808.33	10582.06	10298.93	10323.59	10256.73
25	10541.46	10575.7	10583.62	10421.72	10355.47
30	9444.277	10768.7	8174.46	9268.866	9185.5

ATR	
0.001	11011.48
0.002	10324.45
0.003	10091.54

Take Profi	100	150	200
150	10515.28	10576.73	10613.05
200	10500.77	10529.31	10493.99
250	10403.46	10358.19	10541.23
300	10522.89	10458.36	10564.91

Figure 26: Optimization Results from 2010 to 2019

Figure 26 indicates that the optimized RSI value from 2010 to 2019 would be the lower bound of 35 and upper bound of 65, the optimized stochastic value would be lower bound of 20 and upper bound of 60, and finally the ATR value would be greater than 0.001. The last pivot table is mainly for take profit value. Take Profit value and stop loss are optimized together into one table. It shows that the most profitable take profit value would be 200 and the most profitable stop loss value is 150.

5.3.4 Results

After optimizing, the program is run using the values from the optimization. Table 7 shows the return value from each year from 2010 to 2019.

Table 7: Generated return from year 2010 to 2018

Year	Profit Value
2010 to 2011	18%
2011 to 2012	28%
2012 to 2013	15%
2013 to 2014	23%
2014 to 2015	10%
2015 to 2016	30%
2016 to 2017	21%
2017 to 2018	14%
2018 to 2019	21%

It is gratifying to observe that the program shows a significant and consistent return over almost 9 years. Figure 27 shows how the program would perform if a person were to run the program across 2010 through 2019.

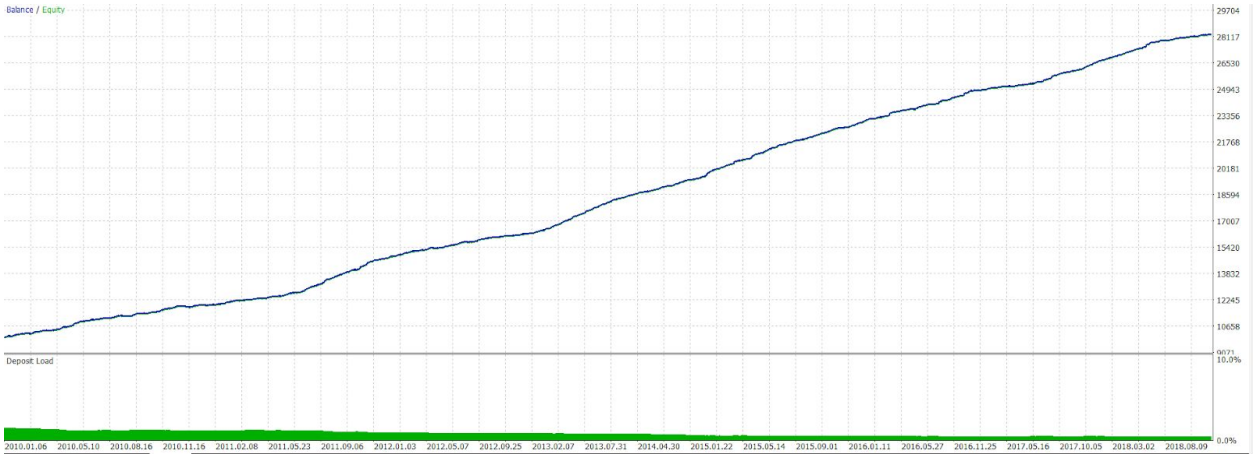


Figure 27: Results from Optimization run across 2010 through 2019

As Figure 27 indicates, after having performed 16846 trades with 9103 short trades and 7743 long trades over nine years there is a net profit of $18270.02. If the total profit were to be divided by nine, it would be approximately $2030 per year. As we started from $10000, the program shows that an average of 20% return from each year. Appendix G shows the complete detail of the trades and the parameters used for the program.

6. Combination of Systems

This Chapter describes the functionality of the combined system of systems including the program methodology, program flow, operating conditions, and backtesting process. The results of the combined system are presented and analyzed to determine system quality.

6.1 Methodology

After creating our own individual systems we combined them into one program. The purpose of creating the system of systems is to determine if the combination performs better, or shows more promise, than the individual systems. By combining the systems, the indicators and trading styles should help to complement each other.

6.2 Program Flow

The combination of systems was coded in Metatrader. Each individual system was incorporated into the final program. In Figure 28 we present the program flow of the combined systems.

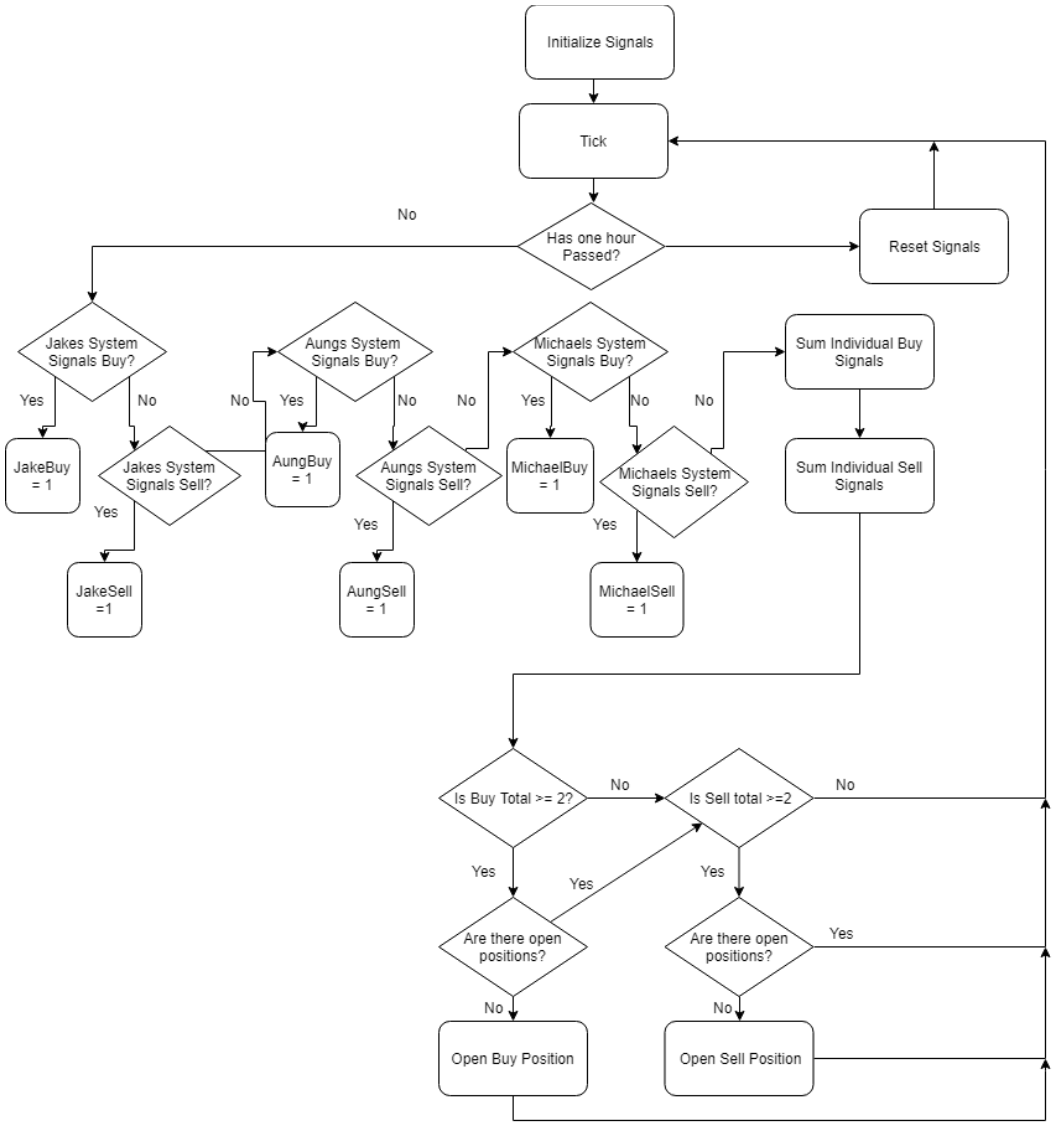


Figure 28: Combined System Program Flow

As seen in Figure 28, the combined system runs in a main loop that is triggered every price tick. Once a tick is detected, the system checks the buy and sell signals of each individual system. Each system was slightly altered to return a string of either “buy” or “sell”, rather than execute a trade. The system works to make a trade if at least two of the systems signal the same trade direction within a one hour time frame. If one hour has passed, the buy and sell signals will

reset. The one hour time frame was chosen to help program functionality. If the combined system only accounts for the current tick signals, the system makes very few trades. Because each system was individually developed the trade conditions occur at different times. With a one hour window, the system is able to make a reasonable number of trades over the selected testing period. The full code for the combined system can be found in Appendix E.

6.3 Operating Conditions

The challenge in designing this system was determining the conditions of program operation. Each individual system had been created to trade with a specific set of conditions, that differed from other systems. To combine the systems we had to design a set of operating conditions that made logistical sense.

The first differing operating conditions were the trade inputs for the size, stop loss, and take profit parameters. For the trade size, we determined the program would trade one lot, or 100% of the initial account balance. This was because two out of the three systems chose to trade using the full initial balance. To determine the take profit and stop loss levels, an approximate average was taken for the three systems. The average was rounded to the nearest 50 pip level. The final result was a stop loss of 150 pips, and a take profit of 400 pips. A trailing stop was also used for this program because a majority of the individual systems utilized a trailing stop.

The other operating conditions related to the Metatrader settings. Each individual system varied slightly with time frame and initial deposit. Figure 29 summarizes the settings used for the combined system.

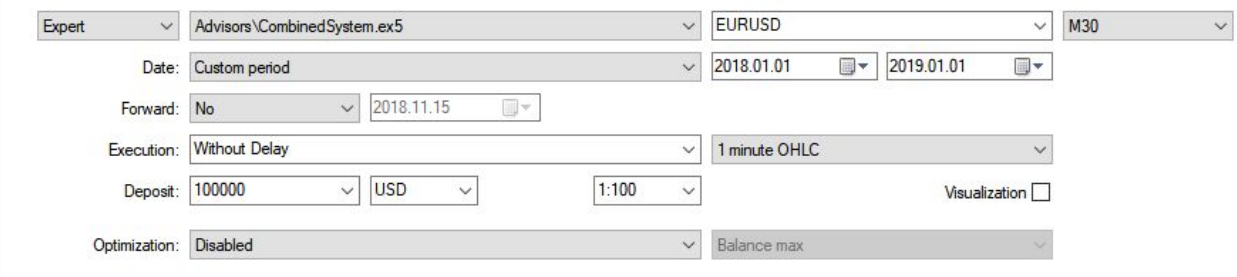


Figure 29: Combined System Settings

Each system was designed to target the EURUSD pair; this was done to maintain consistency. The time frame was determined to be 30 minutes, because a majority of individual systems used this time increment. Lastly, a deposit of $100,000 was employed. The specific amount was not particularly important, but fell in between what individual systems used as initial deposit. More importantly, the trades were made to reflect the usage of the initial deposit, and results were evaluated based on percentage of the initial deposit.

6.4 Results

After the trading conditions were determined, the combined system could be tested on a reasonable time frame. The time selected was January 01, 2010 - January 01, 2019. Clearly, the nine year window was large enough to observe variances in the system, as well as monitor performance in different market conditions. The yearly return for the combined system can be found in Table 8.

Table 8: Combined System Return

Year	Return
2010	3.78%
2011	0.77%
2012	2.81%
2013	1.53%
2014	1.61%
2015	0.64%
2016	3.28%
2017	1.03%
2018	0.76%
Average	1.80%

Over the testing period the combined system returned on average 1.8% a year. Interestingly, there were no years in which the system lost money, but returns were somewhat modest. A variety of additional system measurements can be found in Table 9.

Table 9: Combined System Measurements

Year	Sharpe Ratio	Total Trades	Average Profit	Average Loss	Profit Trade %
2010	0.07	784	0.062%	0.037%	42.09%
2011	0.02	750	0.054%	0.040%	43.87%
2012	0.13	386	0.050%	0.034%	48.96%
2013	0.09	236	0.050%	0.039%	50.85%
2014	0.14	155	0.047%	0.029%	52.26%
2015	0.03	383	0.049%	0.042%	48.04%
2016	0.17	259	0.051%	0.035%	55.60%
2017	0.25	130	0.028%	0.021%	58.46%
2018	0.16	106	0.042%	0.028%	50.00%
Average	0.12	354.33	0.048%	0.034%	50.01%

One concern from this Table is the low average Sharpe ratio. This would imply a high risk for the system, but similar to individual system 1. This may be mitigated by tightening the stop loss value. With a maximum loss of 150 pips each trade can lose $1,500, or 1.5% of the initial deposit. This is very reasonable for a trading system. The system averages close to one trade a day, which is also reasonable for an automated system. It is important to note, that 50.1% of the trades over the testing range were profitable. A majority of the trades were profitable, but the difference between a profit and a loss were quite low, leading to low system return. Additional yearly testing data can be found in Appendix F.

7. Conclusions and Recommendations

This Chapter presents our conclusions and future work recommendations for each of the individual systems, and the combined system. The future recommendations outlines future effort that may improve overall system performance.

7.1 Conclusions

7.1.1 System 1 (Jake Barefoot)

From backtesting this system on historic price data, the approach has the potential to be profitable when used on live data. The 14.46% average yearly return bests popular index funds over that time period. If used with the exact parameters the system was designed for, the system could be used with real money in the future.

Based on the results of the system, it should be more profitable than the most common stock market indexes. The problem is the lack of real time testing, and limited scope of the system. The program was optimized over four years and tested over five years, but there is no live trading results. Popular index funds have been proven to be profitable over the course of 100 years. The system was designed to be used for EURUSD with a 30 minute time frame, and a number of specific constraints. The system runs very well under those conditions, but lacks the robust quality of other proven systems. When run outside the designed conditions, the system does not maintain profitability. To be used as a future investment strategy, the system needs to be able to function under a variety of market conditions.

Nonetheless, the system does accomplish the initial goal of this project. A profitable automated trading system was created that can be easily analyzed. This system serves as a great baseline for future systems, because any of the 15 inputs can be changed to reflect current market conditions. There are a few alterations that need to be made to improve the system quality, but the system does perform well and can serve as a stepping stone for future successful currency trading.

7.1.2 System 2 (Michael Cullen)

This system has the potential to be a very profitable and risk-adverse program. However, as of this writing, the system remains very volatile and follows market trends. So far the program has made profits of over 100% on a few tests, but has also lost up to 99% on others. The most profitable part of the system was simply that roughly 1,500 trades are made each year. It mostly seems that luck drives the program, as some years inexplicably do very well while other years perform very poorly.

However, my goal for this project was not to create a highly profitable system, my goal was to learn the process of foreign exchange and, if possible, learn and develop MQL5 code. After spending weeks studying the system of forex and learning all the different types and functions of various indicators, I set out to learn MQL5. I did not simply develop and test one code, I wrote dozens of codes and tested various indicators, techniques, and parameters on them. To me, this was the best way to learn how to trade forex and it opened the possibility of automatically trading forex in the future.

Also, this project gave me some experience learning a code language and developing a system in that language, as I had no prior coding knowledge. I now feel much more confident in learning an unknown platform and working with it until I understand it well. While I am by no means an expert in MQL5 and I may never use the language again, the fundamentals seem to be very similar to other programming languages.

7.1.3 System 3 (Aung Min)

The program seems to be very promising but it is important to notice that the conditions are very specific. The 20% return arises from the 1 minute OHLC values, and not from each tick; it is based on the 30 minute time-frame. The return is high due to the fact that the years in which the program was tested are the trained years.

This program, listed in Appendix C, can be applied to a real-time trading system if the trader using can set the conditions that the program needs. This program also can be used to backtest and optimize across any year. The user who runs the program should adjust the values within the inputs tabs in Metatrader 5. After changing the inputs, a person can check if his or her trading techniques would work over the years he or she has specified.

Even though this program returns 20% based on the 1 minute OHLC, if this program is executed on every tick, the returns will be inferior. Although the return is high, it is believed that the methodologies can be replicated to work with real tick data. With enough computing power, there is a possibility to obtain every tick based on real ticks and optimize based on all the possible ticks. Unfortunately, due to the lack of computing power and time, this program is

optimized on 1 minute OHLC to reduce the training time and the overhead from the laptop running the optimization.

7.1.4 Combined System

The combined system incorporated the trading strategies of all three individual systems into a single executable program. The final program had an average yearly return of 1.8% from January 2010 to January 2019.

The goal of the project was to create a profitable, automated system of systems that could become a baseline for future trading. We feel that our final program was able to meet this initial goal of the project. Although some of the system profitability metrics were not ideal, the combinational methodology shows promise. The individual systems were designed and optimized on non-standardized operating conditions. Even with the diverging program developments, the combined system was still profitable.

In addition to the profitability of the system of systems, there are other factors that make it a more desirable option than any one individual system. The customization of the system allows future users of the program to alter the inputs for the technical indicators depending on their trading mindset. By combining the various technical indicators and trade parameters used by the individual systems, the system of systems becomes more robust. The indicators can all work together to create a thorough and profitable trading strategy. It is also easy to test, and optimize, in the Metatrader platform. With future adjustments, the program has the potential for high return and consistency.

7.2 Future Recommendations

7.2.1 System 1 (Jake Barefoot)

There are a few improvements that should be made in the future to update the system. The most limiting factors when designing the individual system was time, and computing power. Backtesting and optimization require significant computing power, and multiple hours of run time for each year of optimization. Even using the genetic algorithm, and one minute OHLC settings, each optimization run took approximately 3 hours to complete. Ideally the system should be tested based on real ticks, every combination of inputs, multiple currencies, and multiple time frames. Additional testing and optimization would help to improve reliability of the system under different market conditions, but could not be accomplished with limited computing power and time.

In terms of system program design, the customization and ease of use could be improved. Currently the system runs with a set deposit limit, and does not allow the user to conveniently adjust any of the inputs without modifying the program file. A user interface may be added that would open at program start, allowing the user to specify deposit amount, currency pair being traded, indicator values, etc. Adding this feature would permit anyone to easily interact with the system with no coding knowledge. Allowing users to interact with the program would be a step towards commercialization of the system.

7.2.2 System 2 (Michael Cullen)

Due to computational time and power required to backtest the parameters to the fullest extent, estimations and simplifications were made to gain a working code. Given more time to troubleshoot the code, better parameters may have found. Moreover, after reviewing data from my system, it seems that an additional indicator may have been helpful in limiting the number of bad trades. An RSI or RVI may have helped balance out the losses and make the profit trades more profitable. Although much more testing is required, the platform shows promise.

7.2.3 System 3 (Aung Min)

For future implementations, it would be ideal for the program to be optimized in the parameters of "every tick based on real ticks" with more inputs. The inputs and the ticks within the optimization is reduced greatly in the system above. Also, based on this program, it would be ideal for the program to give out when to buy and sell based on real time data. However, it would be very difficult to follow as it makes tremendous amount of trades each year. Another possible improvement would be to interact with the user via the inputs of optimization. For instance, having input boolean variables such as "use_RSI_indicator" or "use_ATR_indicator" in which the user will have to input "true" or "false" to indicate whether the user wants to use this indicator or not. In the code for the system 3, see Appendix C, the Boolean variables are commented out due to time constraints. With these variables, this system will be more user friendly and will serve as a strategy testing program for novice programmers who want to test their strategies whenever they learn something new.

7.2.4 Combined System

There are future adjustments that should be made for the final program. Initially, the recommendations for each individual system need to be implemented. The combined system functions optimally if all input systems function optimally. Much like the individual systems, there is room for optimization with the combined systems. Future iterations should optimize the combined inputs, and individual system inputs to find the best possible combined system output. It is expected that the optimization for the combined system will use even more resources than individual systems, because of the program complexity, and number of possible inputs.

Once the optimizations are executed for the combined system, and individual systems have been improved, there is also an option for future commercialization. Metatrader and other online trading platforms have a marketplace for trading systems. Users with an account can view information about the program, and may choose to purchase the program for individual use. This could potentially create an additional way to monetize the program in addition to using it for personal forex trading. This decision to post on the Metatrader platform would depend on future improvements. In the systems current state, there might not be high demand for the product, but there is a potential for the system to become more reliable and profitable based on the aforementioned improvements.

References

Achelis, S. B. (2014). Technical analysis from A to Z: Covers every trading tool ... from the absolute breadth index to the zig zag. NY: McGraw-Hill Education.

Bauer, Raimund. MQL5 Tutorial Youtube Channel

Bauer, R. J., & Dahlquist, J. R. (1999). Technical market indicators. New York: Wiley.

Bollinger, J. (1992, February). Using Bollinger Bands. *Stocks & Commodities*, *10*, 47-51.

Chan, L., & Lakonishok, J. (2004). Value and Growth Investing: Review and Update. *Financial Analysts Journal, 60*(1), 71. Retrieved February 25, 2019, from https://www.cfapubs.org/toc/faj/2004/60/1.

Cohen, T. B. (2016). Entry Filtering With Volatility Measures. *IFTA*. Retrieved January 18, 2019, from http://www.ftaa.org.hk/Files/2016.pdf

Double Smoothed Stochastic [Digital Image]. (2019). Retrieved March 25, 2019, from https://www.fidelity.com/learning-center/trading-investing/technical-analysis/technical-indicator-guide/double-smoothed-stochastic

Drakopoulou, V. (2016). A Review of Fundamental and Technical Stock Analysis Techniques. *Journal of Stock & Forex Trading, 05*(01). doi:10.4172/2168-9458.1000163

Fischer, S. (1993). The Role of Macroeconomic Factors in Growth. *The National Bureau of Economic Research, 4565*. Retrieved March 15, 2019, from https://www.nber.org/papers/w4565.

Forex Market Overview. (2019). Retrieved February 25, 2019, from

https://www.nasdaq.com/forex/education/foreign-exchange-market-overview.aspx

Forex trading sessions table [Digital image]. (2017, June). Retrieved February 1, 2019, from http://www.forextips.com/forex-101/forex-market-sessions-trading-times/

Froot, K., & Thaler, R. (1990). Anomalies: Foreign Exchange. *Journal of Economic Perspectives,4*(3), 179-192. Retrieved March 15, 2019, from https://www.aeaweb.org/articles?id=10.1257/jep.4.3.179.

Grimes, A. (2012). The art and science of technical analysis: Market structure, price action, and trading strategies. Hoboken, NJ: Wiley.

King, Michael Robert and Osler, Carol L. and Rime, Dagfinn, Foreign Exchange Market Structure, Players and Evolution (August 14, 2011). Norges Bank Working Paper No. 2011/10, http://dx.doi.org/10.2139/ssrn.1935858

Lo, A. W., Mamaysky, H. and Wang, J. (2000), Foundations of Technical Analysis: Computational Algorithms, Statistical Inference, and Empirical Implementation. The Journal of Finance, 55: 1705-1765.

Major Currency Pairs [Digital image]. (2018). Retrieved February 1, 2019, from gmtraining.weebly.com/lesson-1-section-3-buying-and-selling-in-currency-pairs

MetaQuotes Software Corp. - Page 16. (2019). Retrieved April 10, 2019, from https://www.metatrader4.com/en/company/page16

Moving Average Convergence Divergence [Digital Image]. (2019). Retrieved March 28, 2019, from https://www.investopedia.com/terms/m/macd.asp

MQL5 Articles. (2019). Retrieved February 1, 2019, from https://www.mql5.com/en/articles

Optimization Types, (2019). Retrieved February 5, 2019, from

https://www.metatrader5.com/en/terminal/help/algotrading/optimization_types

Parabolic SAR [Digital Image]. (2011). Retrieved March 25, 2019, from https://www.dailyfx.com/forex/education/trading_tips/daily_trading_lesson/2011/09/30/Parabolic_SAR.html

Patel, P. (2014). Factors affecting Currency Exchange Rate, Economical Formulas and Prediction Models. *International Journal of Application or Innovation in Engineering & Management, 3*(3), 53-56. Retrieved March 15, 2019, from https://www.ijaiem.org/V3I3.html.

Relative Strength Index [Digital Image]. (2019). Retrieved March 25, 2019, from https://www.fidelity.com/learning-center/trading-investing/technical-analysis/technical-indicator-guide/RSI

Strategy Testing. (2019). Retrieved March 20, 2019, from https://www.metatrader5.com/en/terminal/help/algotrading/testing

Triple Exponential Average, (2019). Retrieved February 6, 2019. From https://www.metatrader5.com/en/terminal/help/indicators/oscillators/tea

What are Bollinger Bands? [Digital Image]. (2019). Retrieved March 28, 2019, from https://www.fidelity.com/learning-center/trading-investing/technical-analysis/technical-indicator-guide/bollinger-bands

Appendix A: System 1 Code

The code below is for the system described in Chapter 5.1. The system code was compiled in MetaEditor 5, and was ran in Metatrader 5 to produce the results described.

```
#include<Trade\Trade.mqh>

CTrade trade;

input double SARstep = .03;
input double SARmax = .3;
input int TRIXperiod = 18;
input int MACDfast = 11;
input int MACDslow = 25;
input int MACDsignal = 10;
input int RSIperiod = 8;
input int RSIupper = 60;
input int RSIlower = 40;
input int BANDperiod = 16;
input double BANDsd = 2;
input int TakeP = 400;
input int Stopl = 100;
```

```
input int SLSmover = 350;
input int SLBmover = 250;



void OnTick()
  {

   //Create Price Data
   MqlRates PriceArray[];

   ArraySetAsSeries(PriceArray,true);

   int Data = CopyRates(Symbol(),Period(),0,3,PriceArray);

   double signal = 0;
   double prevsignal = 0;
   double Account_Balance = AccountInfoDouble(ACCOUNT_EQUITY);

   double Ask = NormalizeDouble(SymbolInfoDouble(_Symbol,SYMBOL_ASK),_Digits);
   double Bid = NormalizeDouble(SymbolInfoDouble(_Symbol,SYMBOL_BID),_Digits);
```

```
   //SAR declarations
   double mySARArray[];
   int SARDefinition = iSAR(_Symbol,_Period,SARstep,SARmax);
   ArraySetAsSeries(mySARArray,true);
   CopyBuffer (SARDefinition,0,0,3,mySARArray);
   double SARValue = NormalizeDouble (mySARArray[0],5);
   double LastSARValue = NormalizeDouble(mySARArray[1],5);
   double NextToLastSARValue = NormalizeDouble(mySARArray[2],5);

   //Trix Declarations
   double myMovingTripleAverageArray[];
   int TrixDefinition = iTriX(_Symbol,_Period,TRIXperiod,PRICE_CLOSE);
   ArraySetAsSeries(myMovingTripleAverageArray,true);
   CopyBuffer(TrixDefinition,0,0,3,myMovingTripleAverageArray);

   //MACD Declarations
   double myMACDArray [];
   int MACDDefinition =
iMACD(_Symbol,_Period,MACDfast,MACDslow,MACDsignal,PRICE_CLOSE);
   ArraySetAsSeries (myMACDArray,true);
   CopyBuffer (MACDDefinition,0,0,3,myMACDArray);
```

```
float MACDValue = myMACDArray[0];

//RSI Declarations
double myRSIArray [];
int RSIDefinition = iRSI(_Symbol,_Period,RSIperiod,PRICE_CLOSE);
ArraySetAsSeries(myRSIArray,true);
CopyBuffer(RSIDefinition,0,0,3,myRSIArray);
signal = 0;

//BollingerBands
double MiddleBandArray[];
double UpperBandArray[];
double LowerBandArray[];

ArraySetAsSeries(MiddleBandArray,true);
ArraySetAsSeries(UpperBandArray,true);
ArraySetAsSeries(LowerBandArray,true);

int BollingerBandsDef = iBands(_Symbol,_Period,BANDperiod,0,BANDsd,PRICE_CLOSE);

CopyBuffer(BollingerBandsDef,0,0,3,MiddleBandArray);
CopyBuffer(BollingerBandsDef,1,0,3,UpperBandArray);
```

```
CopyBuffer(BollingerBandsDef,2,0,3,LowerBandArray);


//MACD Buy Conditions
if (MACDValue < 0)
{
  signal = signal + 1;
}


// SAR Buy Conditions
if ((LastSARValue < PriceArray[1].low) && (NextToLastSARValue > PriceArray[2].high))
{
  signal = signal + 1;
}


//Moving Average Buy Conditions
if ((myMovingTripleAverageArray[0] > 0))
{
   signal = signal + 1;
}


//RSI Buy Conditions
if ((myRSIArray[1] <= RSIlower))
```

```
  {
     signal = signal + 1;
  }

  //Bollinger Buy Conditions
  if((PriceArray[1].close < LowerBandArray[1]) && (PriceArray[0].close >
LowerBandArray[0]))
  {
    signal = signal + 1;
  }

  //Sell Conditions


  //MACD Sell Conditions
  if(MACDValue > 0)
  {
    signal = signal - 1;
  }
  //Sell SAR Conditions
  if ((LastSARValue > PriceArray[1].high) && (NextToLastSARValue < PriceArray[2].low))
  {
```

```
    signal = signal - 1;
  }

  //Sell Average Conditions
  if ((myMovingTripleAverageArray[0] < 0))
  {
    signal = signal - 1;
  }

   //Sell Conditions
  if ((myRSIArray[1] >= RSIupper))
  {
    signal = signal - 1;
  }

  //Bollinger Sell Conditions
  if ((PriceArray[1].close > UpperBandArray[1]) && (PriceArray[1].close <
UpperBandArray[0]))
  {
    signal = signal - 1;
  }
```

```
//leave market
if (signal == 0)
  {
    ClosePositions();
  }

//Buy Market
else if (signal >= 3 && PositionsTotal() < 1)
{
  ClosePositions();
  trade.Buy (signal,NULL,Ask,(Bid - Stopl* _Point),(Ask + TakeP* _Point),NULL);
}

//Sell Market
else if (signal <= -3 && PositionsTotal() < 1)
{
  ClosePositions();
  trade.Sell (MathAbs(signal),NULL,Bid,(Ask + Stopl* _Point),(Bid -TakeP * _Point),NULL);
}

CheckTrailingStop(Ask,Bid);
```

```
  signal = 0;
 }

void ClosePositions()
  {
    for (int i=PositionsTotal()-1; i>=0; i--)
      {
      int ticket=PositionGetTicket(i);
      trade.PositionClose(i);
      }
  }

void CheckTrailingStop(double Ask, double Bid)
  {
    double SLB = NormalizeDouble(Ask+SLBmover*_Point,_Digits);
    double SLS = NormalizeDouble(Bid-SLSmover*_Point,_Digits);

    for(int i = PositionsTotal() - 1; i>= 0; i--)
    {
      string symbol = PositionGetSymbol(i);

      if (_Symbol == symbol)
```

```
      {
        ulong PositionTicket = PositionGetInteger (POSITION_TICKET);
        double CurrentStopLoss=PositionGetDouble(POSITION_SL);

        if (CurrentStopLoss < SLB && PositionGetInteger(POSITION_TYPE) ==
POSITION_TYPE_BUY)
        {
          trade.PositionModify(PositionTicket, (CurrentStopLoss + 10 * _Point), 0);
        }

        if (CurrentStopLoss > SLS && PositionGetInteger(POSITION_TYPE) ==
POSITION_TYPE_SELL)
        {
          trade.PositionModify(PositionTicket, (CurrentStopLoss - 10 * _Point), 0);
        }
      }
    }
  }
```

Appendix B: System 2 Code

The code below is for the system described in Chapter 5.2. The system code was compiled in MetaEditor 5, and was ran in Metatrader 5 to produce the results described.

```
#include <Trade\Trade.mqh>
CTrade trade;

input int SMATimeframe=11;
input int LMATimeframe=74;
input int StopLossValue=150;
input int TakeProfitValue=140;

void OnTick()
  {

   MqlRates PriceInfo[];

   ArraySetAsSeries (PriceInfo,true);

   int Data =CopyRates(Symbol(),Period(),0,3,PriceInfo);

   string signal="";

   double ShortMovingAverageArray[];

   double LongMovingAverageArray[];

   double Ask=NormalizeDouble(SymbolInfoDouble(_Symbol,SYMBOL_ASK),_Digits);

   double Bid=NormalizeDouble(SymbolInfoDouble(_Symbol,SYMBOL_BID),_Digits);

   int ShortMovingAverageDefinition = iMA (_Symbol,_Period,SMATimeframe,0,MODE_SMA,
PRICE_CLOSE);

   int LongMovingAverageDefinition = iMA (_Symbol,_Period,LMATimeframe,0,MODE_SMA,
PRICE_CLOSE);

   ArraySetAsSeries (ShortMovingAverageArray,true);

   ArraySetAsSeries (LongMovingAverageArray,true);

   CopyBuffer(ShortMovingAverageDefinition,0,0,3,ShortMovingAverageArray);

   CopyBuffer(LongMovingAverageDefinition,0,0,3,LongMovingAverageArray);

   //---------------------------------------

   if (LongMovingAverageArray[1]>ShortMovingAverageArray[1])
   if (LongMovingAverageArray[2]<ShortMovingAverageArray[2])
      {
         signal="sell";
      }
   if (LongMovingAverageArray[1]<ShortMovingAverageArray[1])
   if (LongMovingAverageArray[2]>ShortMovingAverageArray[2])
      {
         signal="buy";
      }

   if (signal=="buy" && PositionsTotal()<10)
   trade.Buy(0.10,NULL,Ask,(Ask-StopLossValue*_Point),(Ask+TakeProfitValue*_Point),NULL);

   if (signal=="sell" && PositionsTotal()<10)
   trade.Sell(0.10,NULL,Bid,(Bid+StopLossValue*_Point),(Bid-TakeProfitValue*_Point),NULL);

  }
```

Appendix C: System 3 Code

The code below is for the system described in Chapter 5.3. The system code was compiled in MetaEditor 5, and was ran in Metatrader 5 to produce the results described.

```
#include <Trade\Trade.mqh>
CTrade trade;
//create more input into this code for optimization
//input int MyTakeProfitValue = 10;
//input int Stop = 100;
input int SLI = 300;
input int RSI_Upper = 65;
input int RSI_Lower = 35;
input int Stoch_Upper = 80;
input int Stoch_Lower = 15;
input double ATR_Value = 0.0015;
input int SLImover = 100;
input int stoplossvalue = 10;
//input int RSI_period_get;
//input int sto_one, sto_two, sto_three;
//input int Atr_period_get;
//input bool RSI = true;
```

```
//input bool Stoc = true;
//input bool AVR = true;



void OnTick()
  {
  //double CurrentStopLoss = myAP;
  double myAB = AccountInfoDouble(ACCOUNT_BALANCE);
  double myAP = AccountInfoDouble(ACCOUNT_PROFIT);
  double myAE = AccountInfoDouble(ACCOUNT_EQUITY);



   string signal = "";
   MqlRates PriceArray[];

   ArraySetAsSeries (PriceArray, true);



   double Ask = NormalizeDouble(SymbolInfoDouble(_Symbol, SYMBOL_ASK), _Digits);
   double Bid = NormalizeDouble(SymbolInfoDouble(_Symbol, SYMBOL_BID), _Digits);

Comment("\n Ask: ", Ask);
```

```
Comment("\n Bid: ", Bid);


   double KArray[];
   double DArray[];
   double myRSIArray[];
   double mySARArray[];
   double myATRArray[];
   double myLowerBandArray[];
   double myUpperBandArray[];


   ArraySetAsSeries(KArray, true);
   ArraySetAsSeries(DArray, true);
   ArraySetAsSeries(myRSIArray, true);
   ArraySetAsSeries(mySARArray, true);
   ArraySetAsSeries(myLowerBandArray, true);
   ArraySetAsSeries(myUpperBandArray, true);
   ArraySetAsSeries(myATRArray, true);


   int StochasticDefinition = iStochastic(_Symbol, _Period, 5,3,3,MODE_SMA,
STO_LOWHIGH);
   int myRSIDefinition = iRSI(_Symbol, _Period, 14, PRICE_CLOSE);
   int SARDefinition = iSAR(_Symbol, _Period, 0.02, 0.2);
```

```
   int Data= CopyRates(Symbol(), Period(), 0,3, PriceArray);
   int BBDefinition = iBands(_Symbol, _Period, 20,0,2, PRICE_CLOSE);
   int ATRDefinition= iATR(_Symbol, _Period, 14);

   CopyBuffer(StochasticDefinition, 0,0,3, KArray);
   CopyBuffer(StochasticDefinition, 1,0,3,DArray);
   CopyBuffer(myRSIDefinition, 0,0,3, myRSIArray);
   CopyBuffer(SARDefinition, 0,0,3, mySARArray);
   CopyBuffer(BBDefinition, 2,0,3, myLowerBandArray);
   CopyBuffer(BBDefinition, 1,0,3, myUpperBandArray);
   CopyBuffer(ATRDefinition, 0,0,3, myATRArray);

   double KValue0 = KArray[0];
   double DValue0 = DArray[0];
   double myRSIValue = NormalizeDouble(myRSIArray[0],2);
   double LastSARValue = NormalizeDouble(mySARArray[1], 5);  // calculated with 5 digits
behind the point
   double NextToLastSARValue = NormalizeDouble(mySARArray[2], 5);
   double ATRValue = NormalizeDouble (myATRArray[0] , 5);
   double myUpperBandValue = myUpperBandArray[0];
   double myLowerBandValue = myLowerBandArray[0];
   double myLastUpperBandValue = myUpperBandArray[1];
```

```
   double myLastLowerBandValue = myLowerBandArray[1];



   double KValue1 = KArray[1];
   double DValue1 = DArray[1];
   int Stop = 100;
     //Check user based condition here
//   if(myAE >= myAB)
//      trade.Buy(0.10, NULL, Ask, 0,(Ask+MyTakeProfitValue * _Point), NULL);
   //Moving Stop Loss
   // Check Conditions here! There could be something wrong with the PriceArray[1].low
   //if(ATRValue > 0.0001) {
   if(KValue0 < Stoch_Lower && DValue0 < Stoch_Lower && myRSIValue <RSI_Lower &&
ATRValue > ATR_Value /*&& PriceArray[0].close > myLowerBandValue *//*&&
PriceArray[1].close < myLastLowerBandValue */) {
    // if((KValue0 > DValue0) && (KValue1 < DValue1) /*&& (Ask<
myLastLowerBandValue)*/){
      if((KValue0 < DValue0) && (KValue1 > DValue1) /*&& (Bid > myLastUpperBandValue)
*/){
     //Can even check Bolinger Band with Ask and Bid
   //if(KValue0 < 30 && DValue0 < 30 && myRSIValue < 40 && LastSARValue <
PriceArray[1].low) {
```

```
  // if((KValue0 > DValue0) && (KValue1 < DValue1) && (NextToLastSARValue >
PriceArray[2].high) /*&& (CheckSARValue > PriceArray[3].high)*//*&&(CheckSARValue >
PriceArray[0].high)*/) {
   signal = "buy";
   }
   }
   //Check user based condition here
   //Check Leverage
   //if(ATRValue > 0.0001) {
   if(KValue0 > Stoch_Upper && DValue0 > Stoch_Upper&& myRSIValue >RSI_Upper &&
ATRValue > ATR_Value /*&& PriceArray[0].close < myUpperBandValue */ /*&&
PriceArray[1].close > myLastUpperBandValue */ ) {
     //   if((KValue0 < DValue0) && (KValue1 > DValue1) /*&& (Bid >
myLastUpperBandValue) */){
       if((KValue0 > DValue0) && (KValue1 < DValue1) /*&& (Ask<
myLastLowerBandValue)*/){
   //if(KValue0 > 70 && DValue0 > 70 && myRSIValue >60  && LastSARValue >
PriceArray[1].high) {
  //if((KValue0 > DValue0) && (KValue1 < DValue1)&& (NextToLastSARValue >
PriceArray[2].low) /*&& (CheckSARValue > PriceArray[3].low)*/
/*&&(CheckSARValue<PriceArray[0].low)*/) {
   signal = "sell";
```

```
}
}
//}
if(signal == "sell" && PositionsTotal() < 1) {
  //trade.PositionClose();
  trade.Sell(0.10, NULL, Bid, (Ask+Stop *_Point), (Bid-150 * _Point), NULL);
  }

if(signal == "buy" && PositionsTotal() < 1) {
 //  trade.PositionClose();
  trade.Buy(0.10, NULL, Ask, (Bid-Stop *_Point), (Ask+150* _Point), NULL);
  }

    Comment("The current signal is : ", signal);

CheckTrailingStop(Ask, Bid);
  }

 void CheckTrailingStop (double Ask, double Bid) {
 double SLB  = NormalizeDouble(Ask+SLI*_Point, _Digits);
 double SLS  = NormalizeDouble(Bid-SLImover*_Point, _Digits);
 for ( int i = PositionsTotal() -1; i >= 0 ; i-- ) {
```

```
      string symbol = PositionGetSymbol(i);
     if(_Symbol == symbol) {
        ulong PositionTicket = PositionGetInteger(POSITION_TICKET);
        double CurrentStopLoss = PositionGetDouble(POSITION_SL);
        ulong PositionType = PositionGetInteger(POSITION_TYPE);
      if(CurrentStopLoss< SLB && PositionType == POSITION_TYPE_BUY){
        trade.PositionModify(PositionTicket, (CurrentStopLoss+stoplossvalue*_Point),0);
      }
      if(CurrentStopLoss> SLS && PositionType == POSITION_TYPE_SELL) {
        trade.PositionModify(PositionTicket, (CurrentStopLoss-stoplossvalue *_Point),0);
      }
      }
    }
   }
//+------------------------------------------------------------------+
```

Appendix D: System 1 Yearly Results (Jake Barefoot)

The Figures below show the detailed results gathered for the combined system described in Chapter 5.1, and found in Appendix A. Each Figure is a different year of results gathered, and were generated in the Metatrader 5 Strategy tester section.

History Quality	98%				
Bars	11468	Ticks	1347051	Symbols	1
Initial Deposit	1 000 000.00				
Total Net Profit	112 437.70	Balance Drawdown Absolute	0.00	Equity Drawdown Absolute	39.00
Gross Profit	333 959.90	Balance Drawdown Maximal	9 182.10 (0.86%)	Equity Drawdown Maximal	9 260.10 (0.87%)
Gross Loss	-221 522.20	Balance Drawdown Relative	0.86% (9 182.10)	Equity Drawdown Relative	0.87% (9 260.10)
Profit Factor	1.51	Expected Payoff	11.01	Margin Level	18360.24%
Recovery Factor	12.14	Sharpe Ratio	0.13	Z-Score	-0.18 (14.28%)
AHPR	1.0000 (0.00%)	LR Correlation	0.95	OnTester result	0
GHPR	1.0000 (0.00%)	LR Standard Error	8 634.94		
Total Trades	10209	Short Trades (won %)	5144 (47.40%)	Long Trades (won %)	5065 (49.46%)
Total Deals	20418	Profit Trades (% of total)	4943 (48.42%)	Loss Trades (% of total)	5266 (51.58%)
	Largest	profit trade	1 917.00	loss trade	-332.00
	Average	profit trade	67.56	loss trade	-42.07
	Maximum	consecutive wins ($)	10 (842.00)	consecutive losses ($)	16 (-675.00)
	Maximal	consecutive profit (count)	1 917.00 (1)	consecutive loss (count)	-801.00 (7)
	Average	consecutive wins	2	consecutive losses	2

Figure 30: 2014 System 1 Backtesting Data

History Quality	99%				
Bars	11542	Ticks	1377371	Symbols	1
Initial Deposit	1 000 000.00				
Total Net Profit	201 474.60	Balance Drawdown Absolute	3 551.00	Equity Drawdown Absolute	3 650.00
Gross Profit	567 657.90	Balance Drawdown Maximal	5 426.00 (0.54%)	Equity Drawdown Maximal	5 585.00 (0.56%)
Gross Loss	-366 183.30	Balance Drawdown Relative	0.54% (5 426.00)	Equity Drawdown Relative	0.56% (5 585.00)
Profit Factor	1.55	Expected Payoff	18.66	Margin Level	20882.95%
Recovery Factor	36.07	Sharpe Ratio	0.13	Z-Score	0.57 (43.13%)
AHPR	1.0000 (0.00%)	LR Correlation	1.00	OnTester result	0
GHPR	1.0000 (0.00%)	LR Standard Error	5 412.72		
Total Trades	10800	Short Trades (won %)	5503 (50.99%)	Long Trades (won %)	5297 (52.94%)
Total Deals	21600	Profit Trades (% of total)	5610 (51.94%)	Loss Trades (% of total)	5190 (48.06%)
	Largest	profit trade	2 932.00	loss trade	-388.00
	Average	profit trade	101.19	loss trade	-70.56
	Maximum	consecutive wins ($)	16 (4 008.00)	consecutive losses ($)	16 (-552.00)
	Maximal	consecutive profit (count)	4 008.00 (16)	consecutive loss (count)	-1 462.00 (7)
	Average	consecutive wins	2	consecutive losses	2

Figure 31: 2015 System 1 Backtesting Data

History Quality	99%				
Bars	11503	Ticks	1376751	Symbols	1
Initial Deposit	1 000 000.00				
Total Net Profit	113 972.60	Balance Drawdown Absolute	63.00	Equity Drawdown Absolute	84.00
Gross Profit	408 704.00	Balance Drawdown Maximal	4 350.10 (0.40%)	Equity Drawdown Maximal	4 385.10 (0.40%)
Gross Loss	-294 731.40	Balance Drawdown Relative	0.40% (4 350.10)	Equity Drawdown Relative	0.40% (4 385.10)
Profit Factor	1.39	Expected Payoff	10.79	Margin Level	18739.26%
Recovery Factor	25.99	Sharpe Ratio	0.10	Z-Score	0.40 (31.08%)
AHPR	1.0000 (0.00%)	LR Correlation	0.96	OnTester result	0
GHPR	1.0000 (0.00%)	LR Standard Error	8 013.81		
Total Trades	10561	Short Trades (won %)	5405 (53.27%)	Long Trades (won %)	5156 (54.11%)
Total Deals	21122	Profit Trades (% of total)	5669 (53.68%)	Loss Trades (% of total)	4892 (46.32%)
	Largest	profit trade	2 212.00	loss trade	-388.00
	Average	profit trade	72.09	loss trade	-60.25
	Maximum	consecutive wins ($)	15 (1 959.00)	consecutive losses ($)	22 (-756.00)
	Maximal	consecutive profit (count)	3 778.00 (5)	consecutive loss (count)	-1 224.00 (8)
	Average	consecutive wins	2	consecutive losses	2

Figure 32: 2016 System 1 Backtesting Data

History Quality	87%				
Bars	11562	Ticks	1376253	Symbols	1
Initial Deposit	1 000 000.00				
Total Net Profit	81 011.10	Balance Drawdown Absolute	3 741.00	Equity Drawdown Absolute	3 797.00
Gross Profit	335 777.70	Balance Drawdown Maximal	7 465.20 (0.73%)	Equity Drawdown Maximal	7 544.20 (0.74%)
Gross Loss	-254 766.60	Balance Drawdown Relative	0.73% (7 465.20)	Equity Drawdown Relative	0.74% (7 544.20)
Profit Factor	1.32	Expected Payoff	7.65	Margin Level	17272.50%
Recovery Factor	10.74	Sharpe Ratio	0.09	Z-Score	-0.93 (64.76%)
AHPR	1.0000 (0.00%)	LR Correlation	0.84	OnTester result	0
GHPR	1.0000 (0.00%)	LR Standard Error	10 065.51		
Total Trades	10593	Short Trades (won %)	5210 (50.19%)	Long Trades (won %)	5383 (52.29%)
Total Deals	21186	Profit Trades (% of total)	5430 (51.26%)	Loss Trades (% of total)	5163 (48.74%)
	Largest	profit trade	1 629.00	loss trade	-348.00
	Average	profit trade	61.84	loss trade	-49.34
	Maximum	consecutive wins ($)	11 (935.00)	consecutive losses ($)	20 (-3 129.00)
	Maximal	consecutive profit (count)	1 755.00 (5)	consecutive loss (count)	-3 129.00 (20)
	Average	consecutive wins	2	consecutive losses	2

Figure 33: 2017 System 1 Backtesting Data

History Quality	99%				
Bars	11562	Ticks	1369664	Symbols	1
Initial Deposit	1 000 000.00				
Total Net Profit	213 861.30	Balance Drawdown Absolute	0.00	Equity Drawdown Absolute	28.00
Gross Profit	458 167.80	Balance Drawdown Maximal	2 825.00 (0.23%)	Equity Drawdown Maximal	2 953.00 (0.24%)
Gross Loss	-244 306.50	Balance Drawdown Relative	0.23% (2 825.00)	Equity Drawdown Relative	0.24% (2 953.00)
Profit Factor	1.88	Expected Payoff	19.59	Margin Level	20482.32%
Recovery Factor	72.42	Sharpe Ratio	0.21	Z-Score	1.50 (86.64%)
AHPR	1.0000 (0.00%)	LR Correlation	0.96	OnTester result	0
GHPR	1.0000 (0.00%)	LR Standard Error	18 859.98		
Total Trades	10917	Short Trades (won %)	5450 (53.65%)	Long Trades (won %)	5467 (53.98%)
Total Deals	21834	Profit Trades (% of total)	5875 (53.82%)	Loss Trades (% of total)	5042 (46.18%)
	Largest	profit trade	1 288.00	loss trade	-352.00
	Average	profit trade	77.99	loss trade	-48.45
	Maximum	consecutive wins ($)	12 (672.00)	consecutive losses ($)	18 (-594.00)
	Maximal	consecutive profit (count)	2 219.00 (3)	consecutive loss (count)	-624.00 (5)
	Average	consecutive wins	2	consecutive losses	2

Figure 34: 2018 System 1 Backtesting Data

Appendix E: Combined System Code

The code below is for the system described in Chapter 6. The system code was compiled in MetaEditor 5, and was ran in Metatrader 5 to produce the results described.

```
#include<Trade\Trade.mqh>

CTrade trade;

int SLSmover = 350;
int SLBmover = 250;
int SLI = 150;
int SignalBuy = 0;
int SignalSell = 0;

int JakeBuy = 0;
int JakeSell = 0;
int AungBuy = 0;
int AungSell = 0;
int MichaelBuy = 0;
int MichaelSell = 0;
```

```
datetime date1 = TimeCurrent();
datetime date2;

void OnTick()
  {

     //Create Price Data
   MqlRates PriceArray[];

   ArraySetAsSeries(PriceArray,true);

   int Data = CopyRates(Symbol(),Period(),0,3,PriceArray);

   double Account_Balance = AccountInfoDouble(ACCOUNT_EQUITY);

   double Ask = NormalizeDouble(SymbolInfoDouble(_Symbol,SYMBOL_ASK),_Digits);
   double Bid = NormalizeDouble(SymbolInfoDouble(_Symbol,SYMBOL_BID),_Digits);


   string JakeSignal = "null";
   string AungSignal = "null";
```

```
string MichaelSignal = "null";


JakeSignal = JakesSystem(JakeSignal);
AungSignal = AungsSystem(AungSignal);
MichaelSignal = MichaelsSystem(MichaelSignal);


date2 = TimeCurrent();
MqlDateTime str1, str2;
TimeToStruct (date1,str1);
TimeToStruct (date2,str2);


if (str2.hour == str1.hour)
{
  if (JakeSignal == "buy")
     JakeBuy = 1;
  if (AungSignal == "buy")
     AungBuy = 1;
  if (MichaelSignal == "buy")
     MichaelBuy = 1;


  if (JakeSignal == "sell")
     JakeSell = 1;
```

```
    if (AungSignal == "sell")
      AungSell = 1;
    if (MichaelSignal == "sell")
      MichaelSell = 1;

    SignalBuy = JakeBuy + AungBuy + MichaelBuy;
    SignalSell = JakeSell + AungSell + MichaelSell;
  }

  //if ((JakeSignal == "buy" || AungSignal == "buy" || MichaelSignal == "buy") &&
PositionsTotal() < 1)
  if(SignalBuy >= 2 && PositionsTotal() < 1)
  {
    ClosePositions();
    trade.Buy (1,NULL,Ask,(Bid - 150* _Point),(Ask + 400* _Point),NULL);
  }

  //if ((JakeSignal == "sell" || AungSignal == "sell" || MichaelSignal == "sell") &&
PositionsTotal() < 1)
  if (SignalSell >= 2 && PositionsTotal() < 1)
```

```
{
  ClosePositions();
  trade.Sell (1,NULL,Bid,(Ask + 150* _Point),(Bid -400* _Point),NULL);
}

CheckTrailingStop(Ask,Bid);


 if (str2.hour != str1.hour)
{

  JakeBuy = 0;
  JakeSell = 0;
  AungBuy = 0;
  AungSell = 0;
  MichaelBuy = 0;
  MichaelSell = 0;

  date1 = date2;
  SignalBuy = 0;
  SignalSell = 0;
}
```

```
}


string JakesSystem(string JakeSignal)
 {
  double SARstep = .03;
  double SARmax = .3;
  int TRIXperiod = 18;
  int MACDfast = 10;
  int MACDslow = 25;
  int MACDsignal = 10;
  int RSIperiod = 8;
  int RSIupper = 60;
  int RSIlower = 40;
  int BANDperiod = 14;
  double BANDsd = 2;
  int TakeP = 400;
  int Stopl = 100;
  int SLSmover = 350;
  int SLBmover = 250;
  //Create Price Data
```

```
MqlRates PriceArray[];

ArraySetAsSeries(PriceArray,true);

int Data = CopyRates(Symbol(),Period(),0,3,PriceArray);

double signal = 0;
double prevsignal = 0;
double Account_Balance = AccountInfoDouble(ACCOUNT_EQUITY);

double Ask = NormalizeDouble(SymbolInfoDouble(_Symbol,SYMBOL_ASK),_Digits);
double Bid = NormalizeDouble(SymbolInfoDouble(_Symbol,SYMBOL_BID),_Digits);


//SAR declarations
double mySARArray[];
int SARDefinition = iSAR(_Symbol,_Period,SARstep,SARmax);
ArraySetAsSeries(mySARArray,true);
CopyBuffer (SARDefinition,0,0,3,mySARArray);
double SARValue = NormalizeDouble (mySARArray[0],5);
double LastSARValue = NormalizeDouble(mySARArray[1],5);
double NextToLastSARValue = NormalizeDouble(mySARArray[2],5);
```

```
   //Moving Average Declarations
   double myMovingTripleAverageArray[];
   int TrixDefinition = iTriX(_Symbol,_Period,TRIXperiod,PRICE_CLOSE);
   ArraySetAsSeries(myMovingTripleAverageArray,true);
   CopyBuffer(TrixDefinition,0,0,3,myMovingTripleAverageArray);

   //MACD Declarations
   double myMACDArray [];
   int MACDDefinition =
iMACD(_Symbol,_Period,MACDfast,MACDslow,MACDsignal,PRICE_CLOSE);
   ArraySetAsSeries (myMACDArray,true);
   CopyBuffer (MACDDefinition,0,0,3,myMACDArray);
   float MACDValue = myMACDArray[0];

   //RSI Declarations
   double myRSIArray [];
   int RSIDefinition = iRSI(_Symbol,_Period,RSIperiod,PRICE_CLOSE);
   ArraySetAsSeries(myRSIArray,true);
   CopyBuffer(RSIDefinition,0,0,3,myRSIArray);
   signal = 0;
```

```
//BollingerBands
double MiddleBandArray[];
double UpperBandArray[];
double LowerBandArray[];

ArraySetAsSeries(MiddleBandArray,true);
ArraySetAsSeries(UpperBandArray,true);
ArraySetAsSeries(LowerBandArray,true);

int BollingerBandsDef = iBands(_Symbol,_Period,BANDperiod,0,BANDsd,PRICE_CLOSE);

CopyBuffer(BollingerBandsDef,0,0,3,MiddleBandArray);
CopyBuffer(BollingerBandsDef,1,0,3,UpperBandArray);
CopyBuffer(BollingerBandsDef,2,0,3,LowerBandArray);

//MACD Buy Conditions
if (MACDValue < 0)
{
  signal = signal + 1;
}

// SAR Buy Conditions
```

```
  if ((LastSARValue < PriceArray[1].low) && (NextToLastSARValue > PriceArray[2].high))
  {
    signal = signal + 1;
  }

  //Moving Average Buy Conditions
  if ((myMovingTripleAverageArray[0] > 0))
  {
     signal = signal + 1;
  }

  //RSI Buy Conditions
  if ((myRSIArray[1] <= RSIlower))
  {
     signal = signal + 1;
  }

  //Bollinger Buy Conditions
  if((PriceArray[1].close < LowerBandArray[1]) && (PriceArray[0].close >
LowerBandArray[0]))
  {
    signal = signal + 1;
```

```
}

//Sell Conditions



//MACD Sell Conditions
if(MACDValue > 0)
{
  signal = signal - 1;
}
//Sell SAR Conditions
if ((LastSARValue > PriceArray[1].high) && (NextToLastSARValue < PriceArray[2].low))
{
   signal = signal - 1;
}

//Sell Average Conditions
if ((myMovingTripleAverageArray[0] < 0))
{
   signal = signal - 1;
}
```

```
    //Sell Conditions
   if ((myRSIArray[1] >= RSIupper))
   {
      signal = signal - 1;
   }

   //Bollinger Sell Conditions
   if ((PriceArray[1].close > UpperBandArray[1]) && (PriceArray[1].close <
UpperBandArray[0]))
   {
     signal = signal - 1;
   }









   //signal = 3;



   //leave market
/*   if (signal == 0)
```

```
   {
     ClosePositions();
   }*/

 //Buy Market
 else if (signal >= 3 && PositionsTotal() < 1)
 {
   JakeSignal = "buy";
 }

 //Sell Market
 else if (signal <= -3 && PositionsTotal() < 1)
 {
   JakeSignal = "sell";
 }

//  CheckTrailingStop(Ask,Bid);

 return JakeSignal;
 signal = 0;
// Comment ("The current signal is: %g",signal);
//  Comment ("The current volume is: %g",volume);
```

```
 }


string AungsSystem(string AungSignal)
 {

int MyTakeProfitValue = 300;
int SLI = 150;
int RSI_Upper = 65;
int RSI_Lower = 35;
int Stoch_Upper = 80;
int Stoch_Lower = 15;
double ATR_Value = 0.0015;
//input int RSI_period_get;
//input int sto_one, sto_two, sto_three;
//input int Atr_period_get;
//input bool RSI = true;
//input bool Stoc = true;
//input bool AVR = true;
  //double CurrentStopLoss = myAP;
  double myAB = AccountInfoDouble(ACCOUNT_BALANCE);
  double myAP = AccountInfoDouble(ACCOUNT_PROFIT);
```

```
double myAE = AccountInfoDouble(ACCOUNT_EQUITY);



 string signal = "";
 MqlRates PriceArray[];

 ArraySetAsSeries (PriceArray, true);



 double Ask = NormalizeDouble(SymbolInfoDouble(_Symbol, SYMBOL_ASK), _Digits);
 double Bid = NormalizeDouble(SymbolInfoDouble(_Symbol, SYMBOL_BID), _Digits);

Comment("\n Ask: ", Ask);
Comment("\n Bid: ", Bid);

 double KArray[];
 double DArray[];
 double myRSIArray[];
 double mySARArray[];
 double myATRArray[];
 double myLowerBandArray[];
 double myUpperBandArray[];
```

```
   ArraySetAsSeries(KArray, true);
   ArraySetAsSeries(DArray, true);
   ArraySetAsSeries(myRSIArray, true);
   ArraySetAsSeries(mySARArray, true);
   ArraySetAsSeries(myLowerBandArray, true);
   ArraySetAsSeries(myUpperBandArray, true);
   ArraySetAsSeries(myATRArray, true);

   int StochasticDefinition = iStochastic(_Symbol, _Period, 5,3,3,MODE_SMA,
STO_LOWHIGH);
   int myRSIDefinition = iRSI(_Symbol, _Period, 14, PRICE_CLOSE);
   int SARDefinition = iSAR(_Symbol, _Period, 0.02, 0.2);
   int Data= CopyRates(Symbol(), Period(), 0,3, PriceArray);
   int BBDefinition = iBands(_Symbol, _Period, 20,0,2, PRICE_CLOSE);
   int ATRDefinition= iATR(_Symbol, _Period, 14);

   CopyBuffer(StochasticDefinition, 0,0,3, KArray);
   CopyBuffer(StochasticDefinition, 1,0,3,DArray);
   CopyBuffer(myRSIDefinition, 0,0,3, myRSIArray);
   CopyBuffer(SARDefinition, 0,0,3, mySARArray);
   CopyBuffer(BBDefinition, 2,0,3, myLowerBandArray);
```

```
   CopyBuffer(BBDefinition, 1,0,3, myUpperBandArray);
   CopyBuffer(ATRDefinition, 0,0,3, myATRArray);

   double KValue0 = KArray[0];
   double DValue0 = DArray[0];
   double myRSIValue = NormalizeDouble(myRSIArray[0],2);
   double LastSARValue = NormalizeDouble(mySARArray[1], 5);  // calculated with 5 digits
behind the point
   double NextToLastSARValue = NormalizeDouble(mySARArray[2], 5);
   double ATRValue = NormalizeDouble (myATRArray[0] , 5);
   double myUpperBandValue = myUpperBandArray[0];
   double myLowerBandValue = myLowerBandArray[0];
   double myLastUpperBandValue = myUpperBandArray[1];
   double myLastLowerBandValue = myLowerBandArray[1];

   double KValue1 = KArray[1];
   double DValue1 = DArray[1];

     //Check user based condition here
//   if(myAE >= myAB)
//     trade.Buy(0.10, NULL, Ask, 0,(Ask+MyTakeProfitValue * _Point), NULL);
```

```
  //Moving Stop Loss
  // Check Conditions here! There could be something wrong with the PriceArray[1].low
  //if(ATRValue > 0.0001) {
  if(KValue0 < Stoch_Lower && DValue0 < Stoch_Lower && myRSIValue <RSI_Lower &&
ATRValue > ATR_Value /*&& PriceArray[0].close > myLowerBandValue *//*&&
PriceArray[1].close < myLastLowerBandValue */) {
    if((KValue0 > DValue0) && (KValue1 < DValue1) /*&& (Ask<
myLastLowerBandValue)*/){
    //Can even check Bolinger Band with Ask and Bid
  //if(KValue0 < 30 && DValue0 < 30 && myRSIValue < 40 && LastSARValue <
PriceArray[1].low) {
 // if((KValue0 > DValue0) && (KValue1 < DValue1) && (NextToLastSARValue >
PriceArray[2].high) /*&& (CheckSARValue > PriceArray[3].high)*//*&&(CheckSARValue >
PriceArray[0].high)*/) {
  signal = "buy";
  }
  }
  //Check user based condition here
  //Check Leverage
  //if(ATRValue > 0.0001) {
```

```
  if(KValue0 > Stoch_Upper && DValue0 > Stoch_Upper&& myRSIValue >RSI_Upper &&
ATRValue > ATR_Value /*&& PriceArray[0].close < myUpperBandValue */ /*&&
PriceArray[1].close > myLastUpperBandValue */ ) {
      if((KValue0 < DValue0) && (KValue1 > DValue1) /*&& (Bid > myLastUpperBandValue)
*/){
  //if(KValue0 > 70 && DValue0 > 70 && myRSIValue >60  && LastSARValue >
PriceArray[1].high) {
 //if((KValue0 > DValue0) && (KValue1 < DValue1)&& (NextToLastSARValue >
PriceArray[2].low) /*&& (CheckSARValue > PriceArray[3].low)*/
/*&&(CheckSARValue<PriceArray[0].low)*/) {
  signal = "sell";
  }
  }
  //}
  if(signal == "sell" && PositionsTotal() < 1) {
    AungSignal = "sell";
    //trade.Sell(0.10, NULL, Bid, 0, (Bid-150 * _Point), NULL);

    }

  if(signal == "buy" && PositionsTotal() < 1) {
    AungSignal = "sell";
```

```
   //trade.Buy(0.10, NULL, Ask, 0, (Ask+MyTakeProfitValue * _Point), NULL);
   }

     Comment("The current signal is : ", signal);

// CheckTrailStop (Ask);
 return AungSignal;
  }

 void CheckTrailStop (double Ask) {
 double SL  = NormalizeDouble(Ask-150*_Point, _Digits);
// double SLS  = NormalizeDouble(Bid-150*_Point, _Digits);
 for ( int i = PositionsTotal() -1; i >= 0 ; i-- ) {
   string symbol = PositionGetSymbol(i);
  if(_Symbol == symbol) {
      ulong PositionTicket = PositionGetInteger(POSITION_TICKET);
      double CurrentStopLoss = PositionGetDouble(POSITION_SL);
   if(CurrentStopLoss< SL){
      trade.PositionModify(PositionTicket, (CurrentStopLoss+SLI* _Point),0) ;
   }

 }
```

```
}}



string MichaelsSystem(string MichaelSignal)
 {

  int SMATimeframe=14;
  int LMATimeframe=37;
  int StopLossValue=250;
  int TakeProfitValue=280;
  MqlRates PriceInfo[];

  ArraySetAsSeries (PriceInfo,true);

  int Data =CopyRates(Symbol(),Period(),0,3,PriceInfo);

  string signal="";

  double ShortMovingAverageArray[];

  double LongMovingAverageArray[];
```

```
   double Ask=NormalizeDouble(SymbolInfoDouble(_Symbol,SYMBOL_ASK),_Digits);

   double Bid=NormalizeDouble(SymbolInfoDouble(_Symbol,SYMBOL_BID),_Digits);

   int ShortMovingAverageDefinition = iMA
(_Symbol,_Period,SMATimeframe,0,MODE_SMA,PRICE_CLOSE);

   int LongMovingAverageDefinition = iMA
(_Symbol,_Period,LMATimeframe,0,MODE_SMA,PRICE_CLOSE);

   ArraySetAsSeries (ShortMovingAverageArray,true);

   ArraySetAsSeries (LongMovingAverageArray,true);

   CopyBuffer(ShortMovingAverageDefinition,0,0,3,ShortMovingAverageArray);

   CopyBuffer(LongMovingAverageDefinition,0,0,3,LongMovingAverageArray);

   //--------------------------------------

   if (LongMovingAverageArray[1]>ShortMovingAverageArray[1])
```

```
  if (LongMovingAverageArray[2]<ShortMovingAverageArray[2])
    {
      signal="sell";
    }
  if (LongMovingAverageArray[1]<ShortMovingAverageArray[1])
  if (LongMovingAverageArray[2]>ShortMovingAverageArray[2])
    {
      signal="buy";
    }

  if (signal=="buy" && PositionsTotal()<10)
    MichaelSignal = "buy";

//trade.Buy(0.10,NULL,Ask,(Ask-StopLossValue*_Point),(Ask+TakeProfitValue*_Point),NUL
L);

  if (signal=="sell" && PositionsTotal()<10)
    MichaelSignal = "sell";

//trade.Sell(0.10,NULL,Bid,(Bid+StopLossValue*_Point),(Bid-TakeProfitValue*_Point),NULL)
;
```

```
  return MichaelSignal;


 }


void ClosePositions()
  {
    for (int i=PositionsTotal()-1; i>=0; i--)
      {
      int ticket=PositionGetTicket(i);
      trade.PositionClose(i);
      }
  }


void CheckTrailingStop(double Ask, double Bid)
  {
    double SLB = NormalizeDouble(Ask-SLBmover*_Point,_Digits);
    double SLS = NormalizeDouble(Bid-SLSmover*_Point,_Digits);


    for(int i = PositionsTotal() - 1; i>= 0; i--)
    {
      string symbol = PositionGetSymbol(i);
```

```
      if (_Symbol == symbol)
      {
        ulong PositionTicket = PositionGetInteger (POSITION_TICKET);
        double CurrentStopLoss=PositionGetDouble(POSITION_SL);

        if (CurrentStopLoss < SLB && PositionGetInteger(POSITION_TYPE) ==
POSITION_TYPE_BUY)
        {
          trade.PositionModify(PositionTicket, (CurrentStopLoss + 10 * _Point), 0);
        }
        if (CurrentStopLoss > SLS && PositionGetInteger(POSITION_TYPE) ==
POSITION_TYPE_SELL)
        {
          trade.PositionModify(PositionTicket, (CurrentStopLoss - 10 * _Point), 0);
        }
      }
    }
  }
```

Appendix F: Combined System Yearly Results

The Figures below show the detailed results gathered for the combined system described in Chapter 6, and found in Appendix E. Each Figure is a different year of results gathered, and were generated in the Metatrader 5 Strategy tester section.

History Quality	100%				
Bars	12378	Ticks	1452712	Symbols	1
Initial Deposit	100 000.00				
Total Net Profit	3 776.80	Balance Drawdown Absolute	350.00	Equity Drawdown Absolute	372.00
Gross Profit	20 371.00	Balance Drawdown Maximal	1 211.00 (1.17%)	Equity Drawdown Maximal	1 464.00 (1.41%)
Gross Loss	-16 594.20	Balance Drawdown Relative	1.17% (1 211.00)	Equity Drawdown Relative	1.41% (1 464.00)
Profit Factor	1.23	Expected Payoff	4.82	Margin Level	3444.81%
Recovery Factor	2.58	Sharpe Ratio	0.07	Z-Score	-9.43 (99.74%)
AHPR	1.0000 (0.00%)	LR Correlation	0.92	OnTester result	0
GHPR	1.0000 (0.00%)	LR Standard Error	581.66		
Total Trades	784	Short Trades (won %)	780 (42.31%)	Long Trades (won %)	4 (0.00%)
Total Deals	1568	Profit Trades (% of total)	330 (42.09%)	Loss Trades (% of total)	454 (57.91%)
	Largest	profit trade	604.00	loss trade	-134.00
	Average	profit trade	61.73	loss trade	-36.55
	Maximum	consecutive wins ($)	8 (822.00)	consecutive losses ($)	15 (-500.00)
	Maximal	consecutive profit (count)	1 152.00 (3)	consecutive loss (count)	-576.00 (12)
	Average	consecutive wins	3	consecutive losses	4

Figure 35: 2010 Combined System Backtesting Data

History Quality	100%				
Bars	12379	Ticks	1473613	Symbols	1
Initial Deposit	100 000.00				
Total Net Profit	768.60	Balance Drawdown Absolute	1 071.40	Equity Drawdown Absolute	1 193.40
Gross Profit	17 649.00	Balance Drawdown Maximal	1 373.00 (1.35%)	Equity Drawdown Maximal	1 473.00 (1.45%)
Gross Loss	-16 880.40	Balance Drawdown Relative	1.35% (1 373.00)	Equity Drawdown Relative	1.46% (1 467.40)
Profit Factor	1.05	Expected Payoff	1.02	Margin Level	3339.19%
Recovery Factor	0.52	Sharpe Ratio	0.02	Z-Score	-10.67 (99.74%)
AHPR	1.0000 (0.00%)	LR Correlation	0.71	OnTester result	0
GHPR	1.0000 (0.00%)	LR Standard Error	422.59		
Total Trades	750	Short Trades (won %)	744 (43.95%)	Long Trades (won %)	6 (33.33%)
Total Deals	1500	Profit Trades (% of total)	329 (43.87%)	Loss Trades (% of total)	421 (56.13%)
	Largest	profit trade	340.00	loss trade	-164.00
	Average	profit trade	53.64	loss trade	-40.10
	Maximum	consecutive wins ($)	9 (273.00)	consecutive losses ($)	17 (-442.00)
	Maximal	consecutive profit (count)	1 070.00 (7)	consecutive loss (count)	-687.00 (12)
	Average	consecutive wins	3	consecutive losses	4

Figure 36: 2011 Combined System Backtesting Data

History Quality	100%				
Bars	12377	Ticks	1477427	Symbols	1
Initial Deposit	100 000.00				
Total Net Profit	2 810.00	Balance Drawdown Absolute	103.00	Equity Drawdown Absolute	181.00
Gross Profit	9 463.00	Balance Drawdown Maximal	806.00 (0.79%)	Equity Drawdown Maximal	935.00 (0.92%)
Gross Loss	-6 653.00	Balance Drawdown Relative	0.79% (806.00)	Equity Drawdown Relative	0.92% (935.00)
Profit Factor	1.42	Expected Payoff	7.28	Margin Level	3827.10%
Recovery Factor	3.01	Sharpe Ratio	0.13	Z-Score	-6.57 (99.74%)
AHPR	1.0001 (0.01%)	LR Correlation	0.88	OnTester result	0
GHPR	1.0001 (0.01%)	LR Standard Error	357.90		
Total Trades	386	Short Trades (won %)	386 (48.96%)	Long Trades (won %)	0 (0.00%)
Total Deals	772	Profit Trades (% of total)	189 (48.96%)	Loss Trades (% of total)	197 (51.04%)
	Largest	profit trade	309.00	loss trade	-107.00
	Average	profit trade	50.07	loss trade	-33.77
	Maximum	consecutive wins ($)	9 (672.00)	consecutive losses ($)	15 (-303.00)
	Maximal	consecutive profit (count)	782.00 (6)	consecutive loss (count)	-582.00 (10)
	Average	consecutive wins	3	consecutive losses	3

Figure 37: 2012 Combined System Backtesting Data

History Quality	99%				
Bars	12302	Ticks	1464941	Symbols	1
Initial Deposit	100 000.00				
Total Net Profit	1 531.60	Balance Drawdown Absolute	181.40	Equity Drawdown Absolute	213.00
Gross Profit	5 999.00	Balance Drawdown Maximal	854.40 (0.85%)	Equity Drawdown Maximal	1 009.40 (1.00%)
Gross Loss	-4 467.40	Balance Drawdown Relative	0.85% (854.40)	Equity Drawdown Relative	1.00% (1 009.40)
Profit Factor	1.34	Expected Payoff	6.49	Margin Level	3686.89%
Recovery Factor	1.52	Sharpe Ratio	0.09	Z-Score	-2.93 (99.66%)
AHPR	1.0001 (0.01%)	LR Correlation	0.59	OnTester result	0
GHPR	1.0001 (0.01%)	LR Standard Error	284.29		
Total Trades	236	Short Trades (won %)	227 (51.10%)	Long Trades (won %)	9 (44.44%)
Total Deals	472	Profit Trades (% of total)	120 (50.85%)	Loss Trades (% of total)	116 (49.15%)
	Largest	profit trade	539.00	loss trade	-154.00
	Average	profit trade	49.99	loss trade	-38.51
	Maximum	consecutive wins ($)	6 (282.00)	consecutive losses ($)	7 (-212.00)
	Maximal	consecutive profit (count)	1 078.00 (2)	consecutive loss (count)	-462.00 (3)
	Average	consecutive wins	3	consecutive losses	2

Figure 38: 2013 Combined System Backtesting Data

History Quality	98%				
Bars	12297	Ticks	1446246	Symbols	1
Initial Deposit	100 000.00				
Total Net Profit	1 612.20	Balance Drawdown Absolute	310.20	Equity Drawdown Absolute	544.80
Gross Profit	3 787.80	Balance Drawdown Maximal	536.00 (0.53%)	Equity Drawdown Maximal	734.00 (0.73%)
Gross Loss	-2 175.60	Balance Drawdown Relative	0.53% (536.00)	Equity Drawdown Relative	0.73% (734.00)
Profit Factor	1.74	Expected Payoff	10.40	Margin Level	3572.54%
Recovery Factor	2.20	Sharpe Ratio	0.14	Z-Score	-4.82 (99.74%)
AHPR	1.0001 (0.01%)	LR Correlation	0.68	OnTester result	0
GHPR	1.0001 (0.01%)	LR Standard Error	266.91		
Total Trades	155	Short Trades (won %)	147 (52.38%)	Long Trades (won %)	8 (50.00%)
Total Deals	310	Profit Trades (% of total)	81 (52.26%)	Loss Trades (% of total)	74 (47.74%)
	Largest	profit trade	456.90	loss trade	-155.10
	Average	profit trade	46.76	loss trade	-29.40
	Maximum	consecutive wins ($)	9 (381.00)	consecutive losses ($)	6 (-345.00)
	Maximal	consecutive profit (count)	913.80 (2)	consecutive loss (count)	-345.00 (6)
	Average	consecutive wins	3	consecutive losses	3

Figure 39: 2014 Combined System Backtesting Data

History Quality	99%				
Bars	12389	Ticks	1478408	Symbols	1
Initial Deposit	100 000.00				
Total Net Profit	643.00	Balance Drawdown Absolute	190.00	Equity Drawdown Absolute	209.00
Gross Profit	9 094.00	Balance Drawdown Maximal	1 286.00 (1.26%)	Equity Drawdown Maximal	1 674.00 (1.64%)
Gross Loss	-8 451.00	Balance Drawdown Relative	1.26% (1 286.00)	Equity Drawdown Relative	1.64% (1 674.00)
Profit Factor	1.08	Expected Payoff	1.68	Margin Level	4191.05%
Recovery Factor	0.38	Sharpe Ratio	0.03	Z-Score	-7.35 (99.74%)
AHPR	1.0000 (0.00%)	LR Correlation	0.56	OnTester result	0
GHPR	1.0000 (0.00%)	LR Standard Error	378.54		
Total Trades	383	Short Trades (won %)	373 (48.79%)	Long Trades (won %)	10 (20.00%)
Total Deals	766	Profit Trades (% of total)	184 (48.04%)	Loss Trades (% of total)	199 (51.96%)
	Largest	profit trade	313.00	loss trade	-158.00
	Average	profit trade	49.42	loss trade	-42.47
	Maximum	consecutive wins ($)	10 (742.00)	consecutive losses ($)	11 (-237.00)
	Maximal	consecutive profit (count)	742.00 (10)	consecutive loss (count)	-1 254.00 (8)
	Average	consecutive wins	3	consecutive losses	3

Figure 40: 2015 Combined System Backtesting Data

History Quality	99%				
Bars	12451	Ticks	1490108	Symbols	1
Initial Deposit	100 000.00				
Total Net Profit	3 279.80	Balance Drawdown Absolute	481.00	Equity Drawdown Absolute	506.00
Gross Profit	7 353.80	Balance Drawdown Maximal	971.00 (0.97%)	Equity Drawdown Maximal	1 037.00 (1.03%)
Gross Loss	-4 074.00	Balance Drawdown Relative	0.97% (971.00)	Equity Drawdown Relative	1.03% (1 037.00)
Profit Factor	1.81	Expected Payoff	12.66	Margin Level	4389.72%
Recovery Factor	3.16	Sharpe Ratio	0.17	Z-Score	-4.84 (99.74%)
AHPR	1.0001 (0.01%)	LR Correlation	0.84	OnTester result	0
GHPR	1.0001 (0.01%)	LR Standard Error	488.73		
Total Trades	259	Short Trades (won %)	251 (54.98%)	Long Trades (won %)	8 (75.00%)
Total Deals	518	Profit Trades (% of total)	144 (55.60%)	Loss Trades (% of total)	115 (44.40%)
	Largest	profit trade	468.60	loss trade	-157.00
	Average	profit trade	51.07	loss trade	-35.43
	Maximum	consecutive wins ($)	12 (1 032.00)	consecutive losses ($)	8 (-247.00)
	Maximal	consecutive profit (count)	1 032.00 (12)	consecutive loss (count)	-445.00 (5)
	Average	consecutive wins	3	consecutive losses	3

Figure 41: 2016 Combined System Backtesting Data

History Quality	86%				
Bars	12414	Ticks	1477684	Symbols	1
Initial Deposit	100 000.00				
Total Net Profit	1 026.00	Balance Drawdown Absolute	24.00	Equity Drawdown Absolute	62.00
Gross Profit	2 144.00	Balance Drawdown Maximal	259.00 (0.26%)	Equity Drawdown Maximal	392.00 (0.39%)
Gross Loss	-1 118.00	Balance Drawdown Relative	0.26% (259.00)	Equity Drawdown Relative	0.39% (392.00)
Profit Factor	1.92	Expected Payoff	7.89	Margin Level	4208.98%
Recovery Factor	2.62	Sharpe Ratio	0.25	Z-Score	-2.84 (99.55%)
AHPR	1.0001 (0.01%)	LR Correlation	0.86	OnTester result	0
GHPR	1.0001 (0.01%)	LR Standard Error	108.68		
Total Trades	130	Short Trades (won %)	130 (58.46%)	Long Trades (won %)	0 (0.00%)
Total Deals	260	Profit Trades (% of total)	76 (58.46%)	Loss Trades (% of total)	54 (41.54%)
	Largest	profit trade	93.00	loss trade	-58.00
	Average	profit trade	28.21	loss trade	-20.70
	Maximum	consecutive wins ($)	9 (187.00)	consecutive losses ($)	5 (-185.00)
	Maximal	consecutive profit (count)	236.00 (4)	consecutive loss (count)	-185.00 (5)
	Average	consecutive wins	3	consecutive losses	2

Figure 42: 2017 Combined System Backtesting Data

History Quality	99%				
Bars	12406	Ticks	1469992	Symbols	1
Initial Deposit	100 000.00				
Total Net Profit	762.00	Balance Drawdown Absolute	326.00	Equity Drawdown Absolute	453.00
Gross Profit	2 225.00	Balance Drawdown Maximal	391.00 (0.39%)	Equity Drawdown Maximal	553.00 (0.55%)
Gross Loss	-1 463.00	Balance Drawdown Relative	0.39% (391.00)	Equity Drawdown Relative	0.55% (553.00)
Profit Factor	1.52	Expected Payoff	7.19	Margin Level	4002.98%
Recovery Factor	1.38	Sharpe Ratio	0.16	Z-Score	-2.24 (97.49%)
AHPR	1.0001 (0.01%)	LR Correlation	0.70	OnTester result	0
GHPR	1.0001 (0.01%)	LR Standard Error	210.99		
Total Trades	106	Short Trades (won %)	106 (50.00%)	Long Trades (won %)	0 (0.00%)
Total Deals	212	Profit Trades (% of total)	53 (50.00%)	Loss Trades (% of total)	53 (50.00%)
	Largest	profit trade	139.00	loss trade	-81.00
	Average	profit trade	41.98	loss trade	-27.60
	Maximum	consecutive wins ($)	8 (691.00)	consecutive losses ($)	8 (-378.00)
	Maximal	consecutive profit (count)	691.00 (8)	consecutive loss (count)	-378.00 (8)
	Average	consecutive wins	3	consecutive losses	3

Figure 43: 2018 Combined System Backtesting Data

Appendix G: System 3 Backtesting Data

Figure 44 shows the detailed results gathered for the individual system described in Chapter 5.3, and found in Appendix C. The Figure represents the testing data after testing the system code from 2010-2019.

Results

History Quality:	97%				
Bars:	111 393	Ticks:	13231 131	Symbols:	1
Total Net Profit:	18 270.02	Balance Drawdown Absolute:	13.00	Equity Drawdown Absolute:	14.70
Gross Profit:	32 667.26	Balance Drawdown Maximal:	84.30 (0.71%)	Equity Drawdown Maximal:	88.20 (0.74%)
Gross Loss:	-14 397.24	Balance Drawdown Relative:	0.71% (84.30)	Equity Drawdown Relative:	0.74% (88.20)
Profit Factor:	2.27	Expected Payoff:	1.08	Margin Level:	6908.48%
Recovery Factor:	207.14	Sharpe Ratio:	0.23	Z-Score:	1.23 (78.13%)
AHPR:	)001 (0.01%)	LR Correlation:	1.00	OnTester result:	0
GHPR:	)001 (0.01%)	LR Standard Error:	546.28		
Correlation (Profits,MFE):	0.91	Correlation (Profits,MAE):	0.45	Correlation (MFE,MAE):	0.25
Minimal position holding time:	0:00:20	Maximal position holding time:	1:03:39	Average position holding time:	0:02:14
Total Trades:	16 846	Short Trades (won %):	1433 (53.90%)	Long Trades (won %):	5413 (54.33%)
Total Deals:	33 692	Profit Trades (% of total):	9103 (54.04%)	Loss Trades (% of total):	7743 (45.96%)
		Largest profit trade:	121.90	Largest loss trade:	- 9.80
		Average profit trade:	3.59	Average loss trade:	- 1.86
		Maximum consecutive wins ($):	15 (79.30)	Maximum consecutive losses ($):	13 (-19.70)
		Maximal consecutive profit (count):	121.90 (1)	Maximal consecutive loss (count):	-33.40 (10)
		Average consecutive wins:	2	Average consecutive losses:	2

Figure 44: 2010 to 2019 System 3 Backtesting Results

This link has the detailed information about each and every trade made in Chapter 5.3 results section.

https://drive.google.com/open?id=1QLmw7wczT-_IIWxZMq_bFcWQ4xIHKtaN

This link has the complete results for the optimization

https://drive.google.com/open?id=1eFAhwrJpl4VXZ8IANjcRQ9E09QsG2xBK