High Frequency Trading System Design and Process Management

By

Xiangguang Xiao
Bachelor of Engineering, Tianjin University (1996)
Master of Science in Computer Science, Boston University (2006)

Submitted to the System Design and Management Program in Partial Fulfillment of the Requirements for the Degree of

Master of Science in Engineering and Management

At the

Massachusetts Institute of Technology
May 2009

Signature of Author ______________________________
Xiangguang Xiao
System Design and Management Program
May 2009

Certified by ______________________________
Professor Roy E. Welsch
Thesis Supervisor
Professor of Management Science, Statistics, and Engineering Systems
Sloan School of Management and Engineering Systems Division

Certified by ______________________________
Patrick Hale
Director
System Design and Management Program

High Frequency Trading System Design and Process Management

By
Xiangguang Xiao

Submitted to the System Design and Management Program
on May 22, 2009 in Partial Fulfillment of the Requirements for the Degree of Master of Science
in Engineering and Management

ABSTRACT

Trading firms nowadays are highly reliant on data mining, computer modeling and software development. Financial analysts perform many similar tasks to those in software and manufacturing industries. However, the finance industry has not yet fully adopted high-standard systems engineering frameworks and process management approaches that have been successful in the software and manufacturing industries. Many of the traditional methodologies for product design, quality control, systematic innovation, and continuous improvement found in engineering disciplines can be applied to the finance field.

This thesis shows how the knowledge acquired from engineering disciplines can improve the design and processes management of high frequency trading systems. High frequency trading systems are computation-based. These systems are automatic or semi-automatic software systems that are inherently complex and require a high degree of design precision. The design of a high frequency trading system links multiple fields, including quantitative finance, system design and software engineering. In the finance industry, where mathematical theories and trading models are relatively well researched, the ability to implement these designs in real trading practices is one of the key elements of an investment firm's competitiveness. The capability of converting investment ideas into high performance trading systems effectively and efficiently can give an investment firm a huge competitive advantage.

This thesis provides a detailed study composed of high frequency trading system design, system modeling and principles, and processes management for system development. Particular emphasis is given to backtesting and optimization, which are considered the most important parts in building a trading system. This research builds system engineering models that guide the development process. It also uses experimental trading systems to verify and validate principles addressed in this thesis. Finally, this thesis concludes that systems engineering principles and frameworks can be the key to success for implementing high frequency trading or quantitative investment systems.

Thesis Supervisor: **Professor Roy E. Welsch**
Title: Professor of Management Science, Statistics, and Engineering Systems, Sloan School of Management and Engineering Systems Division, Massachusetts Institute of Technology

Acknowledgments

I would like to thank IBM for giving me the opportunity to study at MIT. I owe a debt of gratitude to Masha Maule and Steffi Diamond at IBM for their support and confidence in me.

I especially would like to thank Professor Roy E. Welsch for providing guidance and knowledge. I feel grateful and fortunate to receive his advice and to listen to his wisdom.

The System Design and Management (SDM) faculty and staff have been very helpful throughout the program. I must acknowledge Pat Hale and Bill Foley for being there for me as always. My classmates and friends at MIT have made my experience at MIT an enjoyable, memorable and rewarding one. I learned a great deal from both professors and my fellow classmates.

Finally, I would like to thank my fiancée, Yu Wei for her love, support and advice. Without that support my academic work and thesis would not have been possible.

Table of Contents

Table of Figures

1. Introduction

The financial market is getting more competitive every day. In the trading sector, a competitive edge is razor thin. To achieve even a 50 basis point excess return is often extremely difficult. Trading firms today depend on data mining, computer modeling and software development. Quantitative analysts in the financial industry conduct many day-to-day tasks similar to those in the software and manufacturing industries. Although the term "financial engineering" is well known, there still exists a large gap between financial and engineering disciplines. Quantitative finance has not yet fully adopted the high quality engineering framework and processes that have succeeded in the software and manufacturing industries.

Financial firms that do not value and implement software engineering processes, such as version controls or quality assurance processes, often prove to be ineffective and inefficient in the production and management of their code bases. In academics, quantitative finance classes prefer to teach students mathematical theories and financial modeling. Also, in engineering school, system architecture and product development classes focus on physical products such as a robot design or a new mobile application. This thesis demonstrates that many ideas of product design, quality control, systematic innovation and continuous improvement processes used in systems engineering disciplines can be applied to improving the business of trading firms.

This thesis shows how the processes employed by engineering disciplines can improve the design of high frequency trading. It aims to integrate methods and concepts from various fields, including quantitative finance, system science, software engineering and data mining. Using cases of high frequency trading system design and foreign exchange markets, the findings in this paper can also be applied to non-high frequency trading system design and other financial market segments.

This paper demonstrates how the guiding system design principles and process management, especially backtesting processes, are crucial elements in the development of a successful financial trading system.

1.1. What is a trading system?

Quoting from Wikipedia, "A system is a set of interacting or interdependent entities, real or abstract, forming an integrated whole." The concept of an "integrated whole" can also be expressed in terms of a system embodying a set of relationships that are differentiated from relationships of the set to other elements, and from relationships between an element of the set and elements that are not part of the relational regime. Sharing the same common characteristics, most systems:

- Are abstractions of reality
- Have a structure that is defined by its parts and their composition
- Have behavior, which involves inputs, processing and outputs of material, information or energy
- Have parts with functional as well as structural relationships between each other
- May also refer to a set of rules that govern behavior or structure

A financial trading system is generally considered to be a set of trading rules that control when and at which price you open and close the trade. "A system is simply a plan or set of rules of when to buy and sell securities". (Charles D. Kirkpatrick, Julier R. Dahlquist. *Technical Analysis*) An example of a rule would be "buy when a moving average crosses above another". Variables are the quantities used in the rules – two moving averages, and the parameters are the actual values used in the variables – 5 minutes and 15 minutes. "A system lets us determine a priori how it will react to particular market situations." In an ever-challenging financial market, a trading system is more complex than just a set of trading rules. The system must also include the means of controlling the risks of losing capital, mathematical models, historic data, and sometimes specialized software. A system is an "integrated whole" that consists of key elements, relationship between internal elements and relationships with external elements. A U.S. military strategist, John A. Warden III, introduced the Five Ring System model in *The Air Campaign* contending that any complex system could be broken down into five concentric rings. Each ring - Leadership, Processes, Infrastructure, Population and Action Units - could be used to isolate key elements of any system that needed change. The model was used effectively by Air

Force planners in the First Gulf War. Using Warden's book as a model, I will break down a trading system into the following levels:

- Investment philosophy
- Trading strategies and mathematical models
- Risk management and money Management
- Disciplined processes
- Backtest and Optimization
- Financial data and trading software

Consider each level of system or "ring" as one of the trading system's centers of gravity. The principal idea behind the six rings is to develop a trading system that earns a desired profit in a targeted time frame. To accomplish this goal, we need to address each of the rings individually and collectively to build a robust and profitable trading system. The system developer needs to engage as many rings as possible with more emphasis on the inner rings.

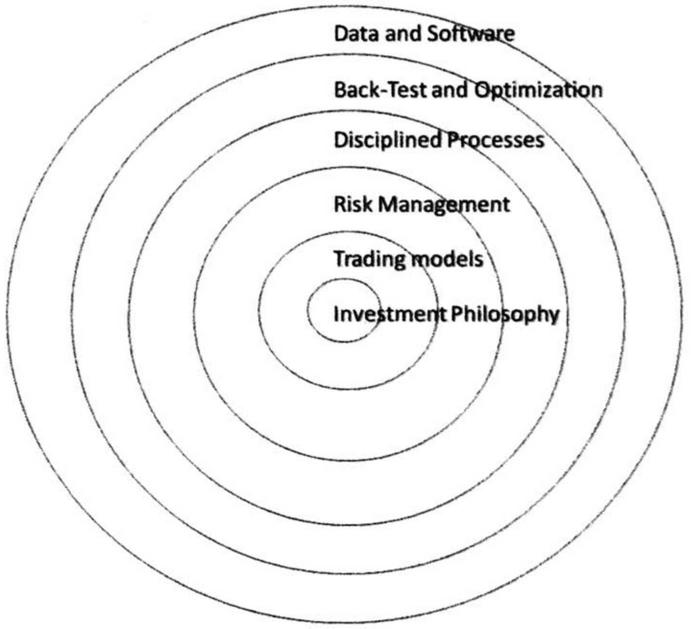


Figure 1: Six Rings of a Trading System

1.2. Why are systems important?

Among the various reasons that traders employ trading systems, the ability to backtest for hypothetical performance using historical data is a key reason. Quantitative funds particularly like the fact that they can quickly validate their new trading ideas. Although there can be a significant difference between hypothetical and actual results, the backtest result can give the financial engineers some idea how their trading systems would work in reality. If the trading system is properly built, financial engineers can leverage the optimization and feedback process to refine their investment ideas, and eventually improve the trading systems.

Another important reason that traders utilize trading systems is that it helps to reduce the emotions involved in making difficult trading decisions. The human emotions of greed and fear sometimes prevent human beings from making timely decisions in the financial market. Trading systems provide a disciplined way to overcome mistakes that can be caused by greed and fear. With a well-designed trading system, you can expect consistency in trading execution and repeatable trading results.

Arguably, a trading system is more important than profit models. Many traders believe that simple and well known trading models can work with successful implementations – good risk control and emotion control. Conceivably, a trading system that implements a naive model could make profits if it includes a solid risk management strategy and reacts to the market in a timely manner.

For high frequency trading firms that depend on the "instant" analysis of data ticks, the use of a trading system is the only way to implement successful investment strategies. Using a trading system fully engages the computing power of IT systems, saving time and reducing human mistakes. Many high frequency trading firms hold the philosophy of "do not predict the market, react to the market". Realizing this philosophy requires a mechanical trading system that captures short time market movements and makes corresponding decisions rapidly.

1.3. High frequency financial data

The original form of market data is tick data which shows every price change and volume. Finance markets generate enormous tick data every trading day. "The number of observations in one single day of a liquid market is equivalent to the number of daily data within 30 years." (Michel M. Dacorogna, Ramazan Gençay, Ulrich Müller, Richard B. Olsen, Olivier V. Pictet. 2001) With advances in information technology, large amounts of high frequency financial data including intraday tick data have become more available. We can now analyze financial time series and markets at almost any desired frequency. Many traders make their trading decisions based upon high-frequency data such as one-minute bar[1] data or even tick-by-tick data. "High frequency data can provide a desirable experimental bench for practitioners to understand market microstructure and for analyzing financial markets." (Dacorogna et al., 2001)

Processing large amounts of data offers both opportunities and challenges for a trading system designer. Statistically, larger amounts of available data result in higher degrees of freedom and bring a new level of significance.[2] From the backtest perspective, more sample trades are generated in the backtest using shorter time scales with the same models. However, challenges also come with the large data size. High frequency data is noisy which can cause problems for modeling as the model may reflect the noise as opposed to the underlying market dynamics. Dealing with large amounts of high frequency data requires the right mathematical tools, models and techniques, which the system designer has to consider when developing their high frequency trading system and backtesting strategies.

The foreign exchange (FX) market is the largest and most liquid financial market in the world. It is attractive for high frequency trading because of its high liquidity, small spreads and low

[1] A bar is a graphical representation of a financial instrument's movement that usually contains the open, high, low and closing prices for a set period of time.

[2] A thorough discussion of high-frequency data can be found in Michel M. Dacorogna, Ramazan Gençay, Ulrich Müller, Richard B. Olsen, Olivier V. Pictet. *An introduction to High-Frequency Finance*. San Diego: Academic Pres. 2001

transaction cost. Since 2008, I have worked with a few other students on building high frequency trading systems on the foreign exchange. This thesis will use the FX market in most studies to leverage the research we have done and the lessons we have learned.

1.4. Contributions of this thesis

1.4.1. Trading system design framework and processes

In the financial market, extensive research is conducted to develop sophisticated mathematical trading models. Equally important, if not more, is the framework of implementation - the design of the production system surrounding the existing financial research/models. The design and processes is the linkage of best practices in engineering disciplines and quantitative finance. In today's trading world, "the key determinant of sustainable competitive advantage is the ability to continually discover, build, and operate better trading systems." (Andrew Kumiega and Benjamin Van Vliet, 2008) Converting an investment idea into mathematical models and then a working trading system in a fast and high quality manner is critical for one to compete in the market place. This thesis identifies the guiding principles and processes for a high frequency trading system design with emphasis on the quality of design and the robustness of the system.

1.4.2. Backtesting and optimization strategies

Among all the components, backtesting and optimization without overfitting are the most important parts in building a trading system. Often misunderstood and misused, backtest and optimization with right strategies to avoid overfitting in the processes are the keys to building a successful trading system. Many trading systems backed by great investment ideas are abandoned during the backtesting phase as they were not tested properly. Other strategies/models with good backtesting performance failed miserably in live trading and caused a huge capital loss because they were improperly optimized. Many financial engineers argue that backtesting and optimization are more important than the mathematical and financial theories behind the trading strategies when building high frequency trading systems. You can build a good trading system with a very simple and straightforward model if backtested well and optimized to control risks properly. With inappropriate testing and optimization, you can convert even a good trading model into a capital disaster. This thesis discusses effective testing and optimization methods to avoid over-fitting with backtesting and optimization.

2. Literature reviews

2.1. *Discretionary versus nondiscretionary systems*

Trading systems can be discretionary and non-discretionary (mechanical), sometimes both. Discretionary trading systems require traders to determine entries and exits, by intuition, or by their own judgment of the importance of technical or fundamental signals that they receive. The number of signals is potentially unlimited. So in a discretionary trading system, traders exercise some discretion in making trades. Nondiscretionary trading systems operate mechanically. Entries and exists are determined mechanically by the system as opposed to traders. The trading decisions are based upon a fixed number of pre-defined technical or fundamental signals without the participation of the trader. Discretionary trading systems require the knowledge and experience from individual traders who must constantly apply their creativity under changing market conditions. A non-discretionary trading system, however, requires creativity from the trader and system designer only in the system design and development phase.

Discretionary trading systems are best used by highly experienced traders with an abundance of practical market knowledge that lets them determine the validity and true meaning of market signals. These traders typically have intuition and have internalized a large number of different historical patterns that they can compare to the current market conditions. They generally have to maintain their mental power and self-discipline to perform consistently under different situations. In essence, discretionary traders develop their brains as a natural trading system. Nondiscretionary traders, however, often take highly systematic approaches. They build the seasoned market insights and knowledge of traders into a working strategy in a software program format. A fully nondiscretionary trading system is one that runs on its own. This type of trading system reads the live market data which is continually fed in by brokers and makes trading decisions automatically based on the pre-determined rules. Trading system designers tend to have an engineering background — good at modeling and programming and familiar with statistics and systems. These designers study the market, read research papers and build models and test their trading ideas using statistical methods. In many firms, financial engineers work with skillful programmers on system design and development and they leverage each other's strengths. The developed trading system usually has been fully backtested on both in-sample and out-of-sample data before it is put into live trading. Starting a mechanical trading system in production without full backtesting is considered to be highly risky. Having a mechanical trading system in place that adheres to a well built plan can prevent haphazard emotional trading. A nondiscretionary trading system enforces discipline. Without strict adherence to a proven nondiscretionary trading system, human emotions can enter into the trading decisions and cause unquantifiable errors. The comparison between discretionary and nondiscretionary systems was performed by Kirkpatrick and Dahlquist in their book *Technical Analysis* (FT press, 2008) and summarized in this section. Authors in this book fully discussed the benefits of nondiscretionary systems. "Researchers have showed that the majority of successful traders and investors use nondiscretionary systems" (Kirkpatrick and Dahlquist, *2008*). "A mechanical approach to the markets can be successful and this is backed up by the fact that approximately 80% of the $30 billion in the managed futures industry is traded by exact systematic methods" (John R. Hill, George Pruitt, and Lundy Hill. *The Ultimate Trading Guide, 2000*).

2.2. What is a good trading system

There are common characteristics of good trading systems. In *Beyond Technical Analysis (Wiley 2001),* Tushar Chande, the author, discusses "Six Cardinal" Rules in trading system design. Kirkpatrick and Dahlquist, in their book *Technical Analysis* (FT press, 2008), consider these rules as the characteristics of a good trading system:

- Positive expectation
- Small number of trading rules - ten rules or less

- Robust parameter values, usable over many different time periods and markets. Being able to trade on different markets is a great indication of trading system robustness.
- Able to trade multiple contracts
- Use risk control, money management, and portfolio design
- Fully mechanical.

These characteristics are regarded as common criteria of a good trading system. However, most trading systems are not evaluated individually. The later sections of this thesis consider trading systems from other perspectives such as their portfolio diversification potential, the liquidity they require and the investment capacity they can provide.

2.3. Simple system versus complex system

Many traders believe that simple trading systems with fewer parameters give comparably good performance and are more robust than the complex trading systems that have many parameters. In *The Alchemy of Trading Systems* (BookSurge Publishing, 2005), Matthew Hanson, the author, explains why simple trading systems work better. With little data, one tends to favor the simpler trading models because they contain fewer parameters and because tests like the likelihood ratio test would strongly penalize the increase of parameters.

In *An introduction to High-Frequency Finance* (Dacorogna et al., 2001), authors had discussion on simplicity versus complexity. "If simplicity is a desirable feature of theoretical models, one should not necessarily seek simplicity at the cost of missing important features of the data-generating process. Sometimes, it is useful to explore more complicated (nonlinear) models that contain more parameters" (Dacorogna et al., 2001). According to the authors, increasing complexity is strongly penalized when explored with low-frequency data because of the loss of degrees of freedom. With high-frequency data, however, the penalty is relatively small because of the abundance of samples even in a limited sampling period thus high-frequency studies can be done for short periods. The results are less affected by structural change in the overall economy than low-frequency studies with samples of many years. This gives us a better opportunity to explore complex trading models. "High-frequency data paves the way for studying financial markets at very different time scales, from minutes to years, and represents an aggregation factor of four to five orders of magnitude." (Dacorogna et al., 2001). Market behaviors observed on low-frequency data can be better validated by the fact that the same behaviors are also observed with high significance on intraday data. This enables the development of more complex trading models, such as those that simultaneously characterize market patterns on different time scales.

Although it is feasible to develop more complex trading systems on high frequency data, it also introduces the risk of overfitting, which increases when factoring in more rules, variables and different timeframes.

2.4. Data sources and data characteristics

High frequency trading is equivalent to high frequency analysis. High frequency trading is more about the analysis of high frequency data than it is about trading frequently.

In *An introduction to High-Frequency Finance* (Dacorogna et al., 2001), the authors provided a thorough discussion of high frequency financial data. A few key points of high frequency data presented by the authors are summarized in the discussion of previous sections (1.3 and 2.3).

Yan and Zivot, in *Analysis of High-Frequency Financial Data with S-Plus (2003)*, also discussed high frequency data characteristics "These high-frequency financial data sets have been widely used to study various market microstructure related issues, including price discovery, competition among related markets, strategic behavior of market participants, and modeling of real time market dynamics. Moreover, high-frequency data are also useful for studying the statistical properties, volatility in particular, of asset returns at lower frequencies."

According to Yan and Zivot (2003), high-frequency financial data possess unique features absent in data measured at lower frequencies:

- *The number of observations in high-frequency data sets can be enormous.* The average daily number of quotes in the EURUSD spot market could easily exceed 20,000. You must handle and analyze a vast amount of data for any high frequency trading system.

- *High frequency data are very noisy.* High-frequency data is noisy and the data needs to be cleaned skillfully before it can be used for modeling.

- *Tick data by nature is irregularly spaced* and with random daily numbers of observations. This tremendously increases the complexity of data cleaning and system development.

The above characteristics of high frequency financial data substantially complicate the process of statistical analysis and trading systems development. This may require specialized statistics and econometrics software packages, more powerful development platforms and rigorous implementation processes.

3. System design and development process

Trading firms can face many obstacles when managing the development and operation of trading systems and the challenges are not unique to financial systems.[3] A trading system in essence is a complex software system. Software engineering is a discipline that leverages systematic and quantifiable approaches to the design, development, operation and reengineering of software systems. Traders and financial analysts can learn to be engineers, although trading models are even less tangible and constructs and stresses are measured in different ways.

[3] Andrew Kumiega, Benjamin Van Vliet, *Quality Money Management*, San Diego: Academic Press. 2008. See quality principles and best practices of managing the development and operation of investment systems in this book.

3.1. Design with Model-Based Systems Engineering (MBSE)

Systems engineering is an interdisciplinary field of engineering that focuses on how complex engineering projects should be designed and managed. A high frequency financial trading system is complex yet needs speed, reliability and robustness. It inherently requires high design precision and design integration. Systems engineering deals with work-processes and tools to handle this kind of project and it overlaps with both technical and human-centered disciplines, such as control engineering and management processes.

Other engineering disciplines are transitioning from a document-based approach to a model-base approach. Friedenthal, Moore and Steiner (A Practical Guide to SysML, 2008) asserts that significant benefits of MBSE are to "enhance communications, specification and design precision, design integration, and reuse that can improve design quality, productivity and reduce the development risk" and MBSE places emphasis on "producing and controlling a coherent system model, and using this model to specify and design the system."

Many MBSE system modeling languages are available. I have used SysML[4] and Object-Process Methodology[5] during my research for high frequency trading system architecture. Both are capable of modeling high frequency trading systems and have different logical processes but serve the same goal: calibrate, bound, and validate your designs.

- SysML is a general purpose graphical modeling language that supports the analysis, specification, design, verification and validation of complex systems. SysML provides a means to capture the system modeling information as part of an MBSE approach without imposing a specific method on how this is performed. The selected method determines which activities are performed, their ordering, and which modeling artifacts are created to represent the system.

---

[4] See more details of SysML in Sanford Friedenthal, Alan Moore, Rick Steiner. *A Practical Guide to SysML: The Systems Modeling Language*. Morgan Kaufmann . 2008

[5] See more details of OPM in Dov Dori , *Object-Process Methodology*. Springer. 2002

- Object-Process Methodology, or OPM, is a new approach to modeling complex systems that consist of humans, physical objects and information. It is a formal paradigm for system development, lifecycle support, and evolution. OPM combines formal yet simple graphics with natural language sentences to express the function, structure, and behavior of systems in an integrated, single model. The name Object-Process methodology comes from its two major building blocks: objects and processes. A third OPM entity is state, which is a situation at which an object can be. Object, processes and states are the only bricks involved in building systems.

SysML is useful for people with a software design background. I found it is easy to describe system-wide processes and function loops in the system design. It is easier to use than OPM when explaining high level processes to financial engineers, traders and other stakeholders who do not necessarily have the knowledge of system architecture. OPM is a great tool that identifies all building blocks (objects) of the system and the internal relationship (process) among these objects. It drills down, zooms in and out so that we can capture the dynamics between different components and sub-systems. OPM may scare you a little with the appearance of complexity, but it is very easy to learn and understand. OPM is especially good for building conceptual models. Both modeling languages are capable of modeling the system architecture and processes of a trading system design. Which system modeling tool is better for you? It depends. You may prefer one to the other, depending on different backgrounds, the preference of thinking processes and the nature of your project. The most important thing is to adopt a system modeling methodology in your trading system and process management. A high quality coherent system model and design framework helps maintain consistency and achieve repeatable success.

Working with Robert Scanlon, a student of system design and management program'09 at MIT, I modeled a high frequency trading system using Object-Process Methodology (OPM). The model was split into several separate diagrams, each representing a process in different levels of the OPM. A high level view of the separate diagrams shows the relationships that exist between them. Each successive diagram is a zoom-in of a process represented in a higher level diagram. All diagrams are in Appendix II for reference. This system model is an experimental model for research purposes and not applicable to building a proper trading system. This thesis uses SysML modeling as the tool to explain system components, processes and design framework. Appendix I has the complete diagram of a trading system design framework and its processes. The purpose of this model is strictly for research and is not applicable to any real world trading systems.

3.1.1. System Components

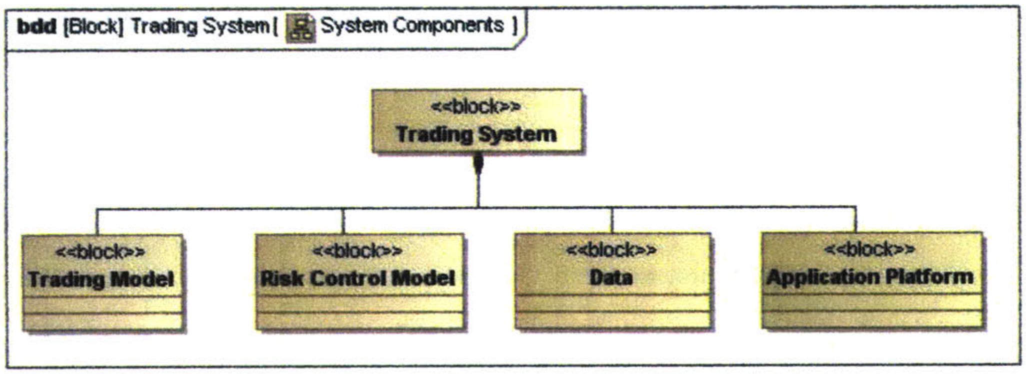


Figure 2: Trading System Components

In Chapter 2.1, a trading system is decomposed into the following six levels:

- Investment Philosophy
- Trading Strategies/Mathematical models
- Risk Management/Money Management
- Disciplined processes
- Backtest and Optimization
- Financial data and trading software

The six levels of a trading system can be absorbed into the system design – risk management, trading strategies, financial data and trading software are components/object of a system and the investment philosophy, backtesting and optimization are processes in the design. Figure 2, following the MBSE approach, displays a first level decomposition:

- Trading strategies/model (the implementation of the investment ideas)

- Data (used to develop trading strategies and backtest)
- Risk Control model
- Application platform (either proprietary software or commercial products)

3.1.2. Users

There are many users directly or indirectly involved in every trading system. The four direct users who are involved into trading system design process are:

Traders, who generally execute trades and may contribute trading ideas and have special domain knowledge that is built into the trading system. Traders are generally the end users of a live trading system. Automatic mechanical trading systems require very minimal human intervention and traders will only monitor the systems and trade by phone as a back-up plan under certain circumstances. Traders may work with programmers to generate a regular basis trade report that contains trading statistics. Later chapters discuss how to measure trading system performance by analyzing trade statistics.

Financial Analysts/Engineers spend most of their time on research and developing strategies, backtesting and optimization. In some firms they are responsible for cleaning data but with help from other teams such as IT support. They primarily focus on algorithm development and prototyping in modeling languages such as SAS, MATLAB, and S-PLUS. They may leave pure technology issues such as error handling and database design for programmers as they usually do not program for production.

Programmers perform production programming. They are skillful with programming in C++, C# or Java. They convert proofed prototypes into real time trading systems. They do not need to understand financial markets, only to create fast, high quality production code. Programmers in have a relatively flexible positions in the trading firm. They may help traders to generate trade reports, or help financial engineers with data preparation.

IT support provides support functions, such as installation and configuration of the trading system and ensuring network stability and security.

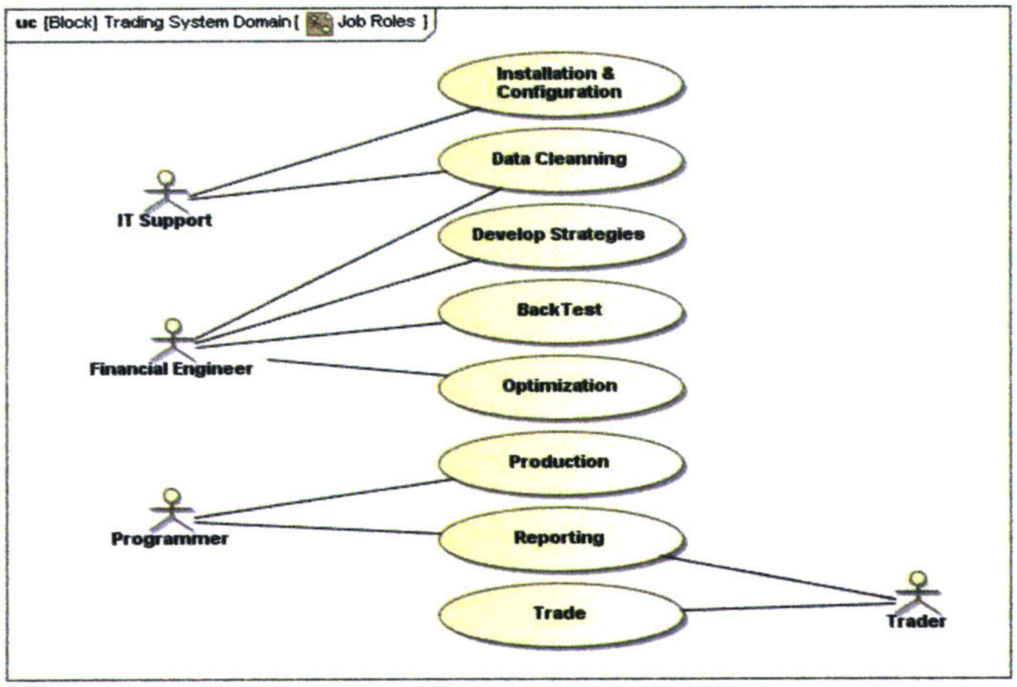


Figure 3: Trading system user cases

Figure 3 describes the major user cases for building and running a trading system in an investment firm. A real firm does not necessarily categorize the team the same way as we do in Figure 3, and job tasks can be overlapped. For instance, financial engineers may code for production in a hedge fund startup, while an established trading firm may even add a separate quality assurance team to test production programs.

3.1.3. Live trading Process

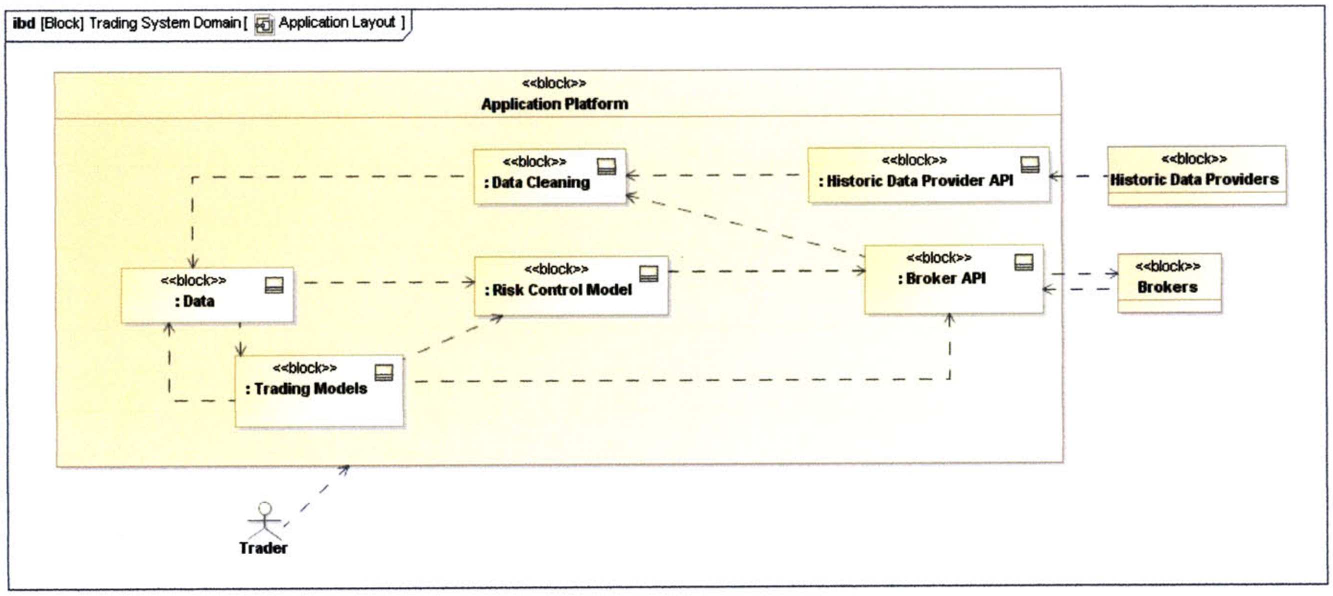


Figure 4: Live trading internal process diagram

The diagram in figure 4 displays a high level overview of the live trading process: the historical data provided by data providers (it can be the firm's proprietary data or the data from live traders) goes through a data cleaning process, which can be used to calculate realized volatility and other market behaviors. The mathematical models of calculation are built into trading models and well tested. Trading models must read the live trading data from the brokers. Usually the live trading data goes through the exact same data cleaning process as for historical data. The live data carries the current market information and the trading model reacts to that information by factoring the quantified historical market behaviors. All trading decisions are evaluated by the risk control model that is also based on historical and live market data. Ultimately, all orders are sent to the Brokers API and executed.

In general, the high-frequency trading system is "a fully-automated process of electronically receiving data, processing that data through decision logic, generating orders, communicating those orders electronically and finally, receiving confirmation of transactions."[6] The trader adds no value in terms of normal operation of the world of systematic trading and investment and

[6] Thomas Neal Falkenberry, *High Frequency Data Filtering*, Tickdata Inc. 2002, [Online]. Available: http://www.tickdata.com/pdf/Tick_Data_Filtering_White_Paper.pdf

automated executions. The trader's job is to monitor and oversee the trading/investment systems. When a machine breaks down and the algorithm stops working, the trader must steps in and decide either to stop trading or process orders manually using a back-up system or by phone. The trader role is critical to the firm.

In addition, Figure 4 shows the components of the application platform, which can be decomposed into its own objects, including the API Module, Data Cleaning Module in live trade, Visualization Module, and Execution Engine. The API Module exists simply to connect the Data Providers to the Trading System. The Data Providers object can be broken into two other objects, Historic Data Providers and Live Data Providers, which each are used as instruments to differentiate sub-processes of Trading.

3.1.1. System Development Process overview

Trading system development can be divided into three major phases: research, prototype and production. Each phase has its own internal processes and sub-systems respectively (See Appendix I):

- *Research phase* – Every trading system starts with its own investment philosophy and trading ideas that are generated by research. There are a variety of method one could use to perform this research. Typically, one generates trading ideas based on market study, literature review, reviewing existing models, or reverse engineering. The outcome of the research phase is a design document of your trading system.

- *Prototype phase* - Prototyping is the phase to verify and validate your investment ideas. In this stage you prepare and partition the historical high frequency data. You also build your models in modeling software, such as MATLAB or S-PLUS. Finally, you backtest your models and optimize them on different data partitions. By analyzing the backtesting performance, you decide to either accept or reject an investment idea. If a trading model is accepted, a detail design specification must be created so that the production team can code precisely.

- *Production phase* – Production is the product realization process of the trading system. It usually follows a software development and quality process and the implementation is based on the detail design specification from the product prototyping. In addition to checking the software quality attributes, paper trading and experimental trading will be performed during the production phase and before launching the live trading service.

Trading system development does not necessarily follow a strict waterfall process. There can be iterations or loops over each design phase as problems arise or new discoveries are made. While Six Sigma or Agile Development may be useful, there is no standard or best approach to implement the trading systems. As in every design project, the approach used and the tools chosen is dependent on the problem, skill of the engineers, development time available, and budget. However, the chosen design approach should provide structure and discipline while being compatible with the abilities of financial engineers and programmers. Failure is most likely to result without such design discipline.

3.1.2. Research phase: generate ideas

The investment philosophy and investment ideas generated by sound research are the foundation for building any investment/trading system. There are various ways to perform this research. Before discussing research approaches, let's look into more fundamental ideas that are the principles and mindset of researching and designing a high frequency trading system:

- *The investment philosophy and investment idea is the foundation* for any trading system. If there is a flaw in our investment philosophy or any logic error with our trading ideas, you are risking capital.

- *Understand the difference between discretionary and nondiscretionary systems*. The high frequency trading system design leans toward a nondiscretionary, automatic system that can be quantified precisely with trading rules and parameters that are explicit and constant.

- *Do not have an opinion of the market.* For a lot of high frequency mechanical trading systems, profits are made from reacting to markets quickly, not by predicting the future of the market.

- *Understand the downside of your investment idea and consider risk management in the research phase.* Start building your risk control model when you begin generating investment ideas.

- *Discipline is a key.* An automatic trading system helps you stay disciplined and away from greed and fear.

- *Develop and customize your own processes.* No existing process fits your research, design and development precisely. Use frameworks and general guidance but customize them and develop your own processes.

- *Test often and without overfitting.* Any research results must be backtested and proved. Test often and avoid overfitting.

3.1.2.1. Research methods

The first decision you make for trading system design is that of trading philosophy and premises built from research. There are various ways to do your investment research. Figure 5 shows that research may include the literature review of academic papers, new perspectives on existing models, market study, and even reverse engineering. "Reverse engineering initiates the redesign process, wherein a product is observed, tracked and analyzed and tested in terms of its performance, trade statistics and other characteristics." (Kumeiga and Van Vliet, 2008) The new design could be a replica of the original or a new adaptation of its underlying trading system(s). Backtesting and optimization will never discover new trading methods. Simply trying different combinations of parameters and trade rules in backtesting typically results in overfitting your strategy to historical data and are destined to fail in live trading.

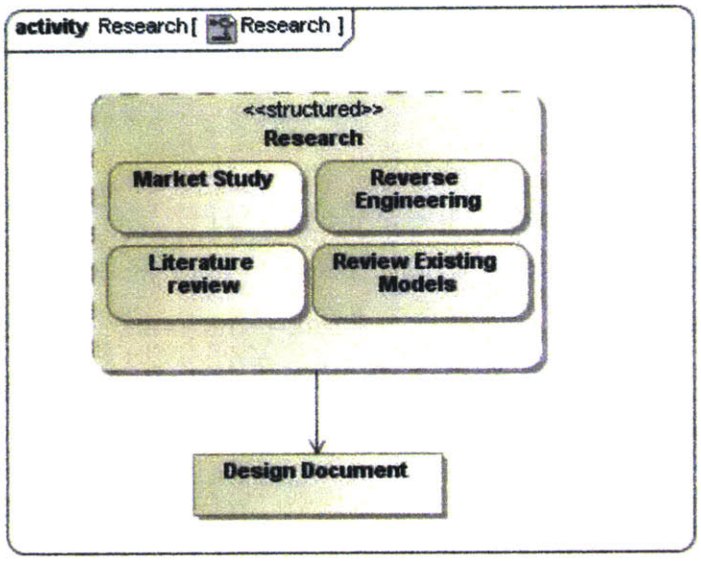


Figure 5: Research phase

3.1.2.2. Design document

The outcome of the research phase is the design document that describes all relevant aspects of our trading ideas. The design document will be used as the blueprint for prototyping in the next phase. The design document can include:

- Description of your investment philosophy and investment ideas
- The intended market
- Specific instruments you will trade on
- How much volatility and liquidity is required
- Trade selection algorithms for both position opening and closing
- Trade execution algorithms
- Data requirements
- Optimization routines
- Time horizon for the system
- Risk management logic
- Performance metrics

- Design alternatives
- Design pitfalls
- Future enhancements

When the design document is created, a team meeting can be held to go through the details. Financial engineers may present their multiple designs in the team meeting and team members can help each other validate the trading philosophy and premises and verify the design. The team meeting acts as a gate meeting – control the quality of the design and decide whether it is good enough to go to the next phase.

3.1.2.3. Case studies

As discussed in 3.1.2.1, there are several approaches to researching and generating trading/investment ideas. For intraday trading strategies, the primary market study approach for is to examine the intra-day patterns of the exchange rate behavior, using the "firm" bid-ask quotes and transactions of currency pairs recorded in the electronic broking system of the spot foreign exchange markets. Once trading ideas emerge, you can document them and generate design documents for prototyping. The following sections discuss two case studies for trading strategies formation.

3.1.2.3.1. Trading the non-farm payroll report

This trading idea originates from the article of "Trading the Non-Farm Payroll Report" by Cory Mitchell (http://www.investopedia.com/articles/forex/09/non-farm-payroll-report.asp). The non-farm payroll (NFP) report is a key economic indicator for the United States. It represents the total number of paid workers in the U.S. exclusive of farm employees, government employees, private household employees and employees of nonprofit organizations.

Of all news announcements, the NFP report consistently causes one of the largest rate movements in the forex market. Upon release, the market is very volatile. Traders usually wait for the wild rate swings to subside, and then attempt to capitalize on the real market move after the speculators have been wiped out or have taken profits or losses. Traders attempt to capture rational movement after the announcement, instead of the irrational volatility that pervades the first few minutes after a news release. The release of the NFP generally occurs on the first Friday of every month at 8:30am EST. This news release creates a favorable environment for active traders in that it provides a near guarantee of a tradable move following the announcement. As with all aspects of trading, whether you make money on it is *not* assured. Approaching the trade from a logical standpoint based on how the market is reacting can provide you with more consistent results than simply anticipating the market movement.

This section expands the outline of the design document as follows:

- Description of your investment philosophy and investment ideas:
    - Of all news announcements, the NFP report consistently causes one of the largest rate movements in the forex market. Trading the NFP report is based on waiting for a small consolidation— the inside bar. After the initial volatility of the report has subsided, the market chooses which direction it will go. By controlling risk with a moderate stop, you are poised to make a potentially large profit from a huge move that almost always occurs each time the NFP is released.
- The intended market
    - Foreign Exchange
- Specific Instruments you will trade on
    - Three currency pairs: USDJPY, USDEUR and GBPUSD
- How much volatility and liquidity is required
    - Realized volatility < 10 basis points. The 10 basis points is the initial default value based on market observation.
    - Accumulated tradable size > 10M US$
- Trade execution algorithms
    - Nothing is done during the first bar after the NFP report (8:30-8:45am in the case of the 15-minute chart).
    - The bar created at 8:30-8:45 will be wide ranging; wait for an inside bar to occur after this initial bar. Wait for the most recent bar's range to be completely inside the previous bar's range.
    - This inside bar's high and low rate set potential trade triggers. When a subsequent bar closes above or below the inside bar, make a trade in the direction of the

breakout. You can also enter a trade as soon as the bar moves past the high or low without waiting for the bar to close.

    - Place a 30 pips[1] stop loss order on the trade you entered.

- Data requirements
    - 15-minutes data from January 2, 2000 to April 30, 2009
    - Economic data
    - Calendar data
- Optimization routines
    - Quarterly
    - Optimize on G7 currency pairs
- Time horizon for the system
    - 15 minutes
- Risk management logic
    - Stop loss order: Place a 30- pips stop loss order on the trade you entered.
    - Time based hard exit: Most of the move occurs within four hours. Exit four hours after your entry time.
    - Maximum trades: Make up to a maximum of two trades. If both get stopped out, do not re-enter. The inside bar's high and low are used again for a second trade, if needed.
    - Leverage: No leverage
- Performance metrics
    - Compounded Total Return
    - Average net return per trade
    - Maximum Intraday Drawdown
    - Sharpe Ratio
    - Sortino Ratio
- Design alternatives
    - Use 5 minutes bar instead
    - A trailing stop is an alternative to exit

**1. A pip is the last decimal point of a quotation. In the Forex market prices are quoted to the fourth decimal point. If the EURUSD moves from 1.4319 to 1.4320, that is one pip.*

3.1.2.3.2. Trading on short term momentum/trend

Trend analysis is the basis for many successful trading programs. Whether these programs are trend following, taking positions in the direction of the trend, or mean reverting, taking positions contrary to price moment, identification of the trend is an essential component. Trend system works because "Prices are not normally distributed but have a fat tail. The fat tail means that there are an unusually large number of directional price moves that are longer than would be expected if prices were randomly distributed." (Perry J. Kaufman. *New Trading Systems and Methods*). The fat tail is a statistical phenomenon caused by a combination of market fundamentals and human behaviors. The net effect is that prices persist in movement in one direction much longer than can be explained by random distribution. Based upon this theory, a "naïve" high frequency strategy can be created – capture the short term momentum/trend in 5 minutes bar. See Figure 6 below. The logic behind this strategy is that whenever the price moves very fast in a short time, there may be a positive feedback of the price action to push the price further.

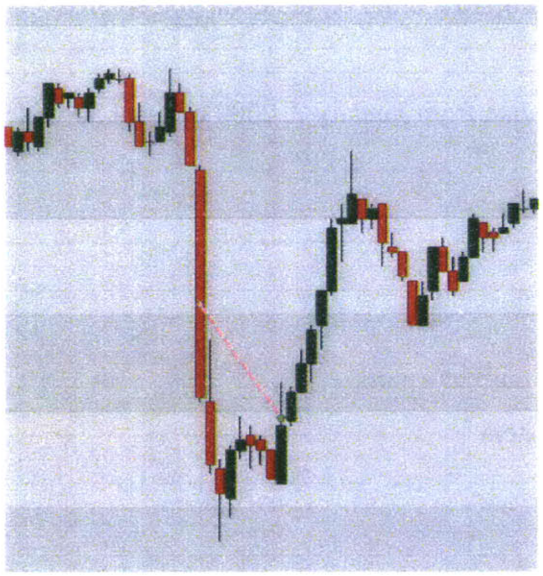


Figure 6: Short term trend

This section is my outline of the design document based upon current thinking:

- Description of your investment philosophy and investment ideas:

    - Whenever the price moves very fast in a short time, there may be a positive feedback of the price action to push the price further. Capture the short term price trend.
- The intended market
    - Foreign Exchange
- Specific Instruments you will trade on
    - USDCHF
- How much volatility and liquidity is required
    - Realized volatility < 12 basis points. The 12 basis points is the initial default value based on market observation.
    - Accumulated tradable size > 15M US$
- Trade execution algorithms
    - Estimate the historic volatility of the past 2 hours by using simple moving average
    - If the historic volatility < 12 basis points, trade is allowed
    - Enter long position when price > 2 hour's high
    - Enter short position when price < 2 hour's low
    - Place a 15-pips stop loss order on the trade you entered
- Data requirements
    - 5-minutes price data from January 2, 2000 to April 30, 2009
    - Economic data
    - Calendar data
- Optimization routines
    - Monthly
    - Optimize on G7 currency pairs
- Time horizon for the system
    - 5 minutes
- Risk management logic
    - Stop loss order: Place a 15-pips stop loss order on the trade you entered.
    - Time based hard exit: Exit 2 hours after the entry time.
    - Maximum trades: One trade at most for each direction at any given time
- Performance metrics
    - Compounded Total Return

  - Average net return per trade
  - Maximum Intraday Drawdown
  - Sharpe Ratio
  - Sortino Ratio
- Design alternatives
  - Use 1-minute bar instead
  - A trailing stop is an alternative to exit
- Implementation Shortfall
  - Stop orders may have big slippage
  - The spread and slippage during the news may be huge, so avoid trade around the economic news release
- Future enhancements
  - Implement an accelerator indicator in order to reduce return volatility - focus not only on the momentum (trend in the underlying), but also on the trend in the momentum signal itself (acceleration).
  - Perform trend survival analysis and statistical calculations to determine if there is a way to find an optimal point (duration or return) to get out of the current position and wait for the next signal.

3.1.3. Prototype phase

Before the prototype, the idea in its abstract format may contain logic errors and flaws in the math. Prototyping proves investment ideas and mathematics before moving on. Prototyping allows you to validate investment philosophy, detect logic errors and evaluate alternative methods and help others understand the math behind investment/trading ideas. The primary reason for prototyping is to resolve uncertainties early in the process. With prototyping, you know whether the mathematical models are feasible and if the system is robust enough to profit under different market conditions. And you can also get a good estimation about the workload of the implementation for live trading applications. As Figure 7 shows, prototyping starts with data preparation and partition, goes through the trading model coding process and finally the backtesting and optimization.

"Prototyping is an exploratory and interactive process." (Kumeiga and Van Vliet, 2008) It follows the requirements of a design document that consists of the investment philosophy, mathematics, data requirements, and so on. The prototyping process forces the team to clearly define the mathematical models and addresses all related issues, such as data issues in the early stage of the process. The result of the prototyping will be a clear definitive requirements specification on which the development of production software will be based. (Kumeiga and Van Vliet, 2008) The document of detail design specification is created in this phase.

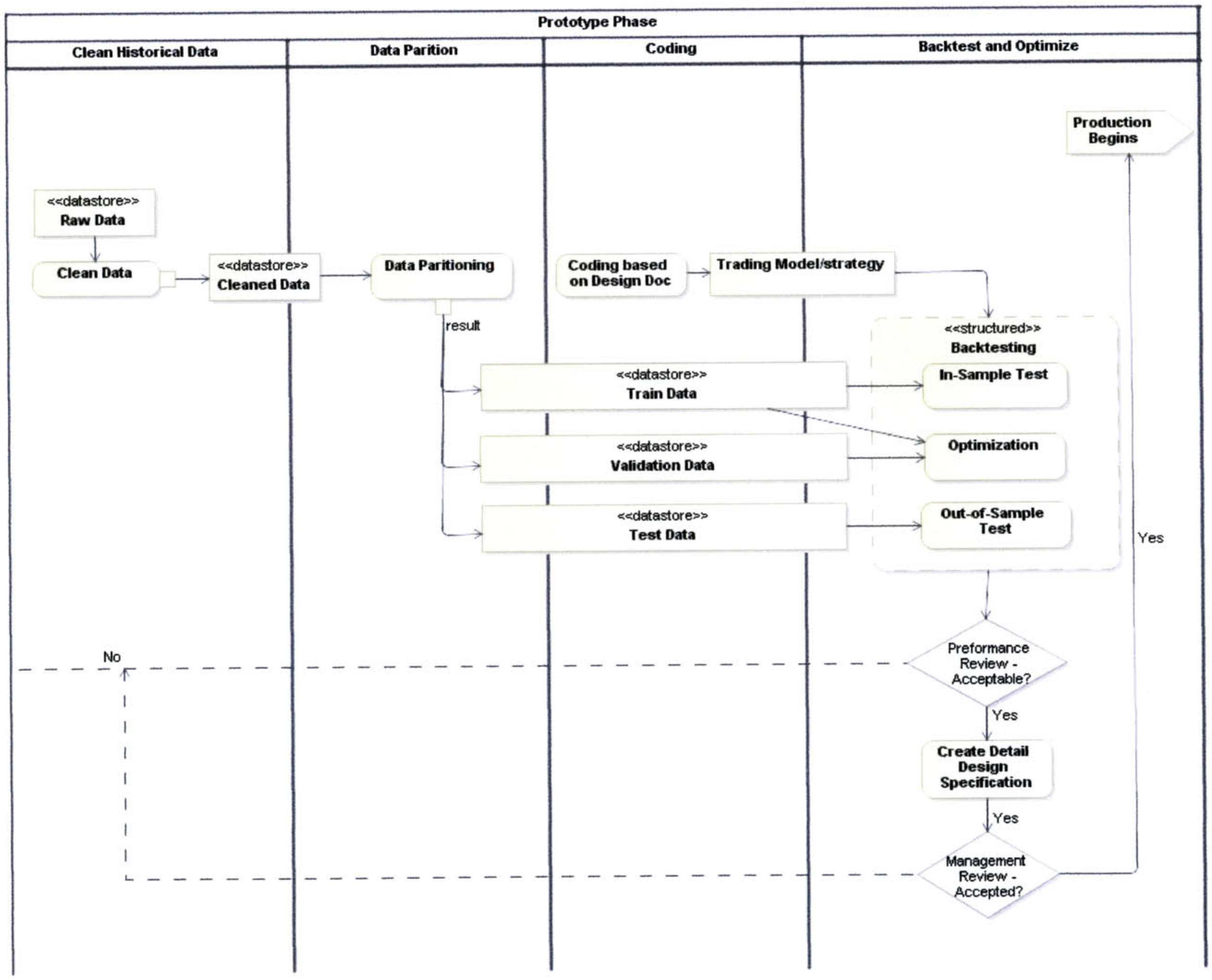


Figure 7: Prototype phase

3.1.3.1. Gather data

The cleaned historical data is the foundation of successful backtesting and optimization. Backtest may itself create problems due to known and unknown errors in the data. Gathering high quality historical data is not just a case of buying and integrating them into the system. For a trading firm, once they decide which asset class they are going to invest or trade, the important first step is to investigate the availability of data needed. They need to decide what kind of data they need, where to get and how to clean the data so that the data fits the needs of their trading systems.

There are three sources for historical data:

- *Purchasing data from historic data providers*, sometimes the historical data can be provided by your brokers if you meet certain criteria. When purchasing data, you need to identify the need for data (tick based, 1-minute based, etc.) and data availability. "You also need to consider the experience of vendors, their references from their past customers, their financial viability and compliance" (Kumeiga and Van Vliet, 2008) Lastly, you must judge data quality independent of the claims of data providers. If you have existing trading systems with some known high quality data, run a backtest on both your data and the data from the vendors and do a comparison.

- *There are free data to download from internet.* Some companies or organizations offer free data downloads. This approach is not recommended since you do not have control of the data quality. Unreliable data can lead to the wrong microstructure calculation resulting in bad trades in which the money lost would be more than the price of the better quality data. Using high quality, more expensive data can be cost efficient. However, even high quality data may have problems so checking data quality and the cleaning process cannot be skipped.

- You can also start collecting data by *recording the market price using programs* and cleaning and converting the data into the formats you really need. Brokers often provide an application programming interface (API) that can feed in live trading data. Some commercial trading platform provide a function called record market data and replay so you can 'save' today's data and test your trading system in the future using today's data.

This approach requires the financial engineers to have sophisticated programming and database administration skill.

Olsen & Associates has a high frequency currency database. NYSE TAQ provides Equity historical high frequency data, and Berkeley Options database provides Options high frequency data. There are also commercial redistributors such as (wrds.wharton.upenn.edu) and QAI Fast-Tick (www.qaisoftware.com)

3.1.3.2. Data type

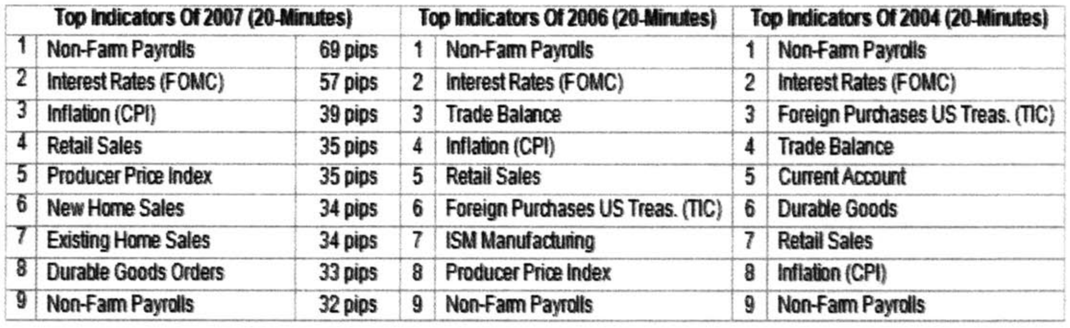
"Sophisticated trading/investment systems may potentially incorporate several different types of data." (Kumeiga and Van Vliet, 2008) Data types include:

- *Price data*[7]usually consists of the read bid/ask price, trade, and volume data. For high frequency trading system, the tick data might be flat ticket which means multiple trades going off at the same price. Some financial derivatives don't have historical trade price data, instead, they use *valuation data*[8].

- *Economic data* tends to be one of the most important factors for short-term market movements and this is particularly true in the currency market. High frequency trading systems are sensitive to short term price movements so economic data is important to the systems. For currency trading, the most important economic releases are: Non-Farm Payrolls, Interest rates, retail sales, inflation (CPI), trade balance and manufacturing PMI. Some economic data is extremely important to short time price movement. The list in Figure 8 ranks the most market-moving data for the U.S. dollar in 2007, 2006 and 2004, on a 20-minute basis.

[7] Definition of Price data is based on *Quality Money Management* (Kumeiga and Van Vliet, 2008). See more details of different data types from this book.

[8] Definition of Valuation data (next page) is based on *Quality Money Management* (Kumeiga and Van Vliet, 2008)

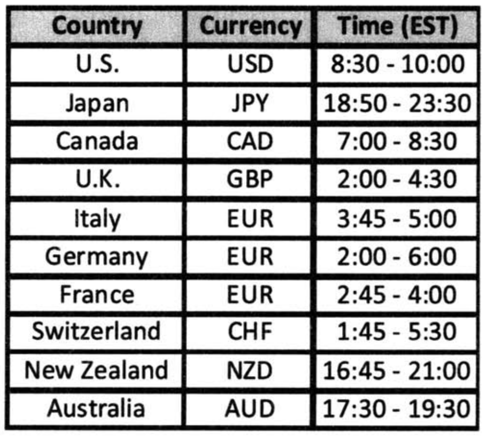


Top Indicators Of 2007 (20-Minutes)			Top Indicators Of 2006 (20-Minutes)		Top Indicators Of 2004 (20-Minutes)	
1	Non-Farm Payrolls	69 pips	1	Non-Farm Payrolls	1	Non-Farm Payrolls
2	Interest Rates (FOMC)	57 pips	2	Interest Rates (FOMC)	2	Interest Rates (FOMC)
3	Inflation (CPI)	39 pips	3	Trade Balance	3	Foreign Purchases US Treas. (TIC)
4	Retail Sales	35 pips	4	Inflation (CPI)	4	Trade Balance
5	Producer Price Index	35 pips	5	Retail Sales	5	Current Account
6	New Home Sales	34 pips	6	Foreign Purchases US Treas. (TIC)	6	Durable Goods
7	Existing Home Sales	34 pips	7	ISM Manufacturing	7	Retail Sales
8	Durable Goods Orders	33 pips	8	Producer Price Index	8	Inflation (CPI)
9	Non-Farm Payrolls	32 pips	9	Non-Farm Payrolls	9	Non-Farm Payrolls

Figure 8: Most influential market moving indicators

Source: John Kicklighter, Top 5 Most Market Moving Indicators for the US Dollar, DailyFx.com, 2008

Calendar data consists of the release date/time of the most important economic data. Some high frequency trading systems are purely built based on price data. Even these systems need to be aware of economic calendars by either avoid trading on that period, or take advantage of it such as "trading on new" strategies. Figure 9 lists the approximate times (EST) at which the most important economic releases for most traded currencies. These are also the times at which you and your trading models should be paying extra attention to the markets.



Country	Currency	Time (EST)
U.S.	USD	8:30 - 10:00
Japan	JPY	18:50 - 23:30
Canada	CAD	7:00 - 8:30
U.K.	GBP	2:00 - 4:30
Italy	EUR	3:45 - 5:00
Germany	EUR	2:00 - 6:00
France	EUR	2:45 - 4:00
Switzerland	CHF	1:45 - 5:30
New Zealand	NZD	16:45 - 21:00
Australia	AUD	17:30 - 19:30

Figure 9: Times of important economic releases

Source: Kathy Lien, Trading On News Releases, Investopedia.com, 2008

- *Valuation data* is different from price data. For over-the-counter (OTC) derivatives, no historical trade price data exists. The price of these assets exists only in theory, so it is a valuation price.

- *Fundamental data* consists of key business items, such as earnings, sales, inventories, and rents. It can be calculated data based on fundamental facts such as P/E Ratio, ROE, and Free Cash Flow etc.

3.1.3.3. Data cleaning process

Algorithm creation, backtesting and risk management requires good and clean data. High frequency tick raw data is noisy, and the data cleaning and filtering process is partially important. The importance of data cleaning is underestimated quite often by financial engineers who might spend days on backtesting before realizing the data is not good enough. Bad data can lead to bad models and losing investment capital. High-frequency trading is data sensitive so spending time on cleaning is vital.

Price data is most often used in high frequency data. A high frequency currency trading system relies on four major sub-processes to clean the price data.

- Check timestamps. Some timestamps are not in the correct order and require sorting. .

- Eliminate outliners. To reduce the effect of data errors and outliners, search and replace/delete bad ticks in the original time series.

- Filter out non-trading days, including weekends, holidays and special events. For foreign exchanges, filter weekend data (5pm Friday - 5pm Sunday).

- Converting into different timeframes such as 1 minute, 5 minutes, 15 minutes, and 30 minutes so traders can use this data based on the requirements of their trading strategies.

According Kirkpatrick and Dahlquist (2008), the historical data vendor should always be consistent with the data vendor of live trading in practice because different vendors may have different data feeding; The authors also pointed out that "dirty" data can happen on a live feed thus data cleaning is also necessary in live trading. So, in practice, the live trading system also has an internal function for data cleaning, as Figure 4 shows.

3.1.3.4. Data partition

When using the entire data set for developing the model, you may introduce overfitting. Given a set of random data, you may be able to create a model that generates good performance. Using the whole data set to build out the model, you may be able to generate a superior model, but in most case you are creating overfitting. To address this problem, use data partitioning – divide cleaned historical data into three partitions that include training data, validation data and test data.

- *Training data* (sometimes called the In-Sample data) typically is the largest partition and contains the data used to build models. The amount of data required depends on the periods of the system. A general rule is that the data set can "generate at least 30 to 50 trades and cover where the market traveled up, down and sideways" (Kirkpatrick and Dahlquist, 2008) After confirming the parameters and trading rules with training data, run optimization on it. Alternatively, re-divide data by following the optimization methods described in Chapter 6.

- *Validation data* is used to assess the performance of each model so you can compare different models. You can also compare the performance on validation data to the one generated based upon training data. If a model with good performance on training data immediately fails with validation data, you know there is overfitting. Use the validation data for fine-tuning to improve the model.

- *Test data* (sometimes called the Out-of-Sample data) is used to re-verify the models and ensure everything is working properly, with no adjustment anymore.

For high frequency systems, the in-sample test is on training data. Optimization is done on both training and validation data.

3.1.3.5. Coding in modeling software

For practitioners, MATLAB, SAS, S-PLUS are great tools for prototype programming. Many financial engineers use Excel, but it may not be powerful enough to handle vast amount of high frequency data. If your models require high computation power, you may have to either upgrade your hardware or have your in-house developed system support the modeling. Usually in-house systems leverage the power of C++ or C# to develop specific software tools to support certain file handling or computation in a much faster manner. However, C++ and C# require higher programming skills compared to MATLAB and other typical modeling languages. There are always decisions and tradeoffs to make when considering different aspects of your project.

In general, prototyping coding is totally separated from production programs. People who perform this task can be in totally different groups. The first group of people can be traders, financial analysts and financial engineers. The prototyping will be based on widely used financial modeling tools, such as MATLAB and SAS. The second group can be software engineers, who do not have any opinions on the models. These people implement the production code in C++ and C# based on the definitive requirements specification generated from the prototyping. The argument is that prototyping focuses on speed and financial engineers need to explore individual requirements, design and implementation options as quickly as possible. Fast exploration is often at the expense of reliability, program performance, robustness and program maintainability, but the whole prototyping process involves many rapid iterations. These sacrifices (the lack of reliability, robustness, etc.) actually enable developers to explore design alternatives and detect design flaws in a cheap way. Another argument to support this approach is that the financial engineers' strength is in the quantitative side and may not have the programming skill to generate high quality production code with the robustness and reliability required in live trading. Conversely, highly experienced programmers may not have sufficient investment knowledge to generate good investment ideas. Some trading firms prefer to have good "pure" programmers who do not have much financial knowledge so "they are able to resist the temptation to change your idea and add some extra 'features' in production". This seems more like an organization management issue to me.

Another approach is called "Evolutionary Prototypes" (Andrew Kumiega, Benjamin Van Vliet. *Quality Money Management*) which means the product team prototypes selected parts of a system first, and then evolves the rest of the system from those parts. This approach does not discard the prototyping code. Rather, it evolves the code into the production application that will be used in live trading. That is, prototyping evolves into the production software. The working system is the final product of a series of evolutionary prototypes. In this approach, prototyping must be built on high quality code from the very beginning. Evolutionary prototyping requires strong management and it works well for a highly experienced and well organized team. It may slow down the process if the team lacks necessary skills. According to Quality Money Management, the evolutionary prototyping should start with riskiest areas. Prototyping the riskiest part first helps you identify the biggest obstacle so you can estimate the efforts and feasibility of the project. This approach may work well with a small agile team with solid skills in both finance and software engineering. Or the project is a cumulative development project. The team starts with a simple working model and gradually builds more comprehensive models on the top the working model in the previous iteration by either modifying/adding new features or integrating other systems.

Modeling language such as MATLAB provide features that convert prototyping codes into C++ or C# format. This kind of feature enables you to reuse the prototyping code in production and facilitates evolutionary prototypes.

3.1.3.6. Prototype quality assurance

During the prototyping phase, the most important aspect of programming quality assurance is to be sure that all calculations are correct. This sounds simple, but it is surprisingly overlooked. Before backtesting and optimization, always ensure the correctness of modeling code.

3.1.3.7. Backtest and Optimization

According to Kumiega and Van Vliet (*Quality Money Management*, 2008), a backtest is a test of the ability of trading/investment strategies to meet the requirements of the design document and "simulation and statistical analysis of a trading/investment strategies' input and output based on cleaned historical data by factoring in transaction costs and slippage in execution." A trading strategy will be accepted and sent to production if it generates acceptable performance, otherwise it will be rejected. For the 'rejected' trading models, either it will be re-investigated at the research phase, or it is discarded for good.

Optimizing is a process to achieve the best result (may not necessarily be the best return) by changing the parameters of a trading strategy/model. Designing experiments is one of the most important benefits of optimization. The designer might find parameters and trading rules that do not work under any circumstance and can be eliminated.

Backtest and optimization can be performed at two stages – the prototyping phase and the production phase. Backtesting and optimization at the prototyping phase is "essentially the make or break stage of the trading/system design and development project." (Kumiega and Van Vliet 2008) The project either gets killed or moves forward to the prototyping stage. The backtesting and optimization at the production phase is a verification and validation process for the quality of the production – the implementation of prototype. The right methodologies and processes of backtesting and optimization are vital and are a focus point of this thesis. Chapters 4 and Chapter 5 will discuss backtesting and optimization respectively.

3.1.4. Production

Once the prototypes of trading and risk models are accepted, production is the last procedure to convert a trading/investment ideas into live trading strategies/models. The detailed design specification from the prototyping phase and the accepted prototypes will be the material the implementation is based on.

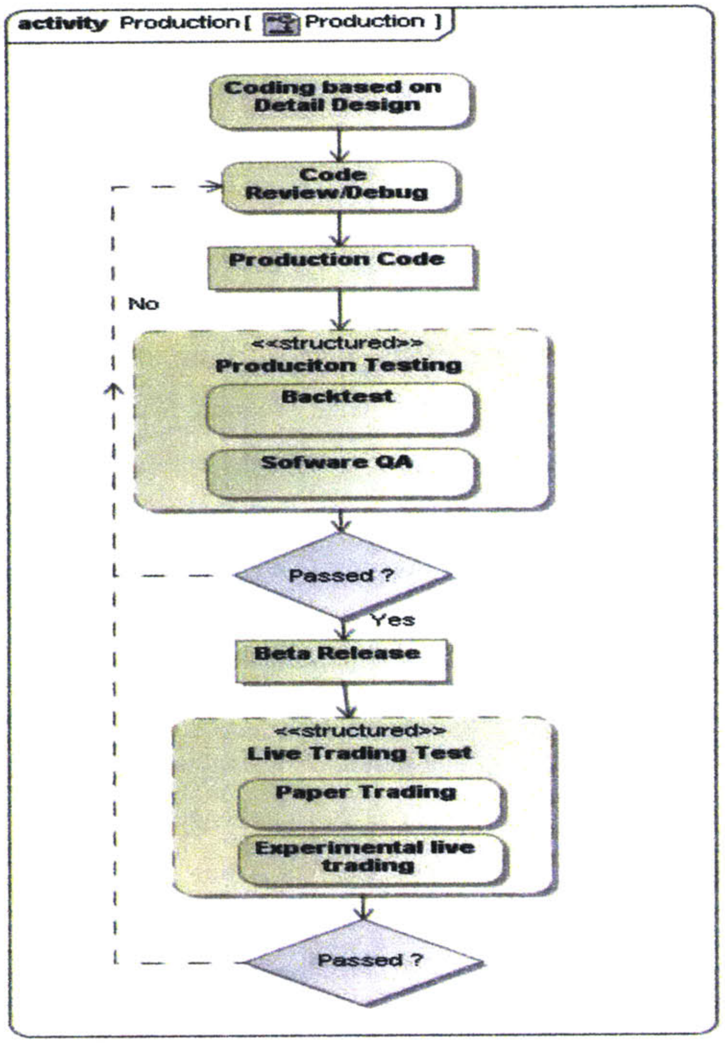


Figure 10: Production Phase

3.1.4.1. *Production programming*

Production programming should follow software engineering processes and the best practices of the software industry. Some companies may depend on highly skilled software engineers for the implementation/coding without a systematic design approach. The assumption is that good engineers can make the design work without any process. Extraordinary people, who are highly experienced and motivated, can get the job done, but dream teams are rare and companies risk burning out their best employees. Heroics in software development are an indication of process failure that can lead to dysfunctional behavior in both organizations and individuals (Yourdon, Edward. *Death March: The Complete Software Developer's Guide to Surviving "Mission Impossible" Projects*, Prentice Hall, Englewood Cliffs, NJ, 1997.) Software engineering practices are designed to make the development of software less chaotic and reliably repeatable.

One of the most important characteristics of a trading system is it must have high quality. No functional bug is allowed. A small error can cause a large capital loss or simply a disaster. Some coding strategies, such as 'code, release and fix later', used by some internet startups will not work with trading firms. You need adopt a suitable programming process and apply the best practices of software engineering to form your own (most comfortable) approach to achieve superior quality of coding. The methodology and practices that were used to code experimental systems of this research are Test-driven development (TDD) and Pair trading.

- TDD is a software development technique that uses short development iterations based on pre-written test cases that define desired improvements or new functions. Each iteration produces the code necessary to pass that iteration's tests. Finally, programmers re-factor the code to accommodate changes. A key TDD concept is that preparing tests before coding facilitates rapid feedback changes (Quoted from Wikipedia).

- Pair programming is a software development technique in which two programmers work together at one work station. One types in code while the other reviews each line of code as it is typed in. The person typing is called the driver. The person reviewing the code is called the navigator. The two programmers switch roles frequently (possibly every 30 minutes). While reviewing, the observer also considers the strategic direction of the work, coming up with ideas for improvements and likely future problems to address. This frees the driver to focus all of his or her attention on the "tactical" aspects of completing the current task, using the observer as a safety net and guide (Quoted from Wikipedia).

At first glance, TDD and Pair programming appears to take longer and cost more. Research, however, has shown the opposite result – these processes and techniques increase discipline, provide better time management, and the coding quality is much improved. Again, there is no standard best approach for production programming. You need to create your own development process that best fits your team.

As Figure 10 shows, besides implementing software quality processes, financial engineers run backtest for the production version of trading system to ensure it achieves the expected performance on historical data. For some cases, financial engineers run optimization against the production code to see if there is any difference between the prototypes and the production.

3.1.4.2. Development team and tool

The competitive advantage of real time high frequency trading system is speed. For some high frequency trading systems, fast computation and execution are critical. Unlike the programming in prototypes, the programming during the production phase shifts from fast iterations that validate mathematical models, to robustness, reliability and fast execution in live trading. Programming tools used will be C++, C# or others that generate fast-execution at runtime. Prototyping usually is done by financial engineers while production is performed by 'pure' programmers with the right skill set.

3.1.4.3. Production quality assurance

For production code, there are three phases of quality assurance. First, the software coding quality has to be assured. Software quality assurance methodologies apply here and can include test-driven development, scenario based testing and test automation. As is true for development process, you can adopt the right approach for software quality assurance, considering the extremely high quality requirement for trading systems. Second, the testing will include the backtesting and sometimes optimization. The major purpose of the backtesting at this stage is not to prove the validity of investment ideas and the math as you do in prototyping. Instead, backtesting compares the performance of the production code to one of the prototypes so you know if the production code was implemented successfully or not. With the same historical data, one would expect similar performance. During certain circumstances, the testing result can be different between prototyping and production. However, the gap should be in an acceptable range and the reason for the gap should be fully investigated. Finally, there is a live trading test before you start the full trading. This test includes:

- Paper trading test. This is simulated trading where a trading system generates real-time signals based on the live data, but only trades with virtual money and executes dummy orders. Paper backtesting is helpful to uncover algorithm failures under extreme market conditions.
- Experimental trading test. This is live trading with small amounts of money. Experimental trading helps you detect algorithm flaws in the live market and helps identify operational issues.

Paper Trading
Experimental Trading
Full live Trading

Figure 11: Three trading types

After evaluating the performance generated by paper trading and experimental trading (performance measurement is discussed in Chapter 6) and deciding to go or kill, the trading system will be either sent back for rework, or be launched for a full live trading. For high frequency trading strategies, it is best to start paper trading first, and then small money experimental trading before final launch of full live trading (see figure 11). For low-frequency data, it may not be feasible to go through all three types of trading because it may wait too long to accumulate enough paper trades and experimental trades that represent a good sample.

4. System backtest

A backtest is a simulation of a trading/investment strategy on historical data. It is a validation and verification process for model assumptions and parameters. It is essentially a quality assurance test against the detailed design document generated in the prototyping phase. After financial engineers determine what data to use, it is then necessary to partition data into test data, validation data, and test data, run an in-sample test, optimize and finally run an out-of-sample test. Financial engineers will then analyze the outputs of tests in order to validate against the design document for the strategy developing. The outcome of tests will help you decide whether to accept the trading strategies or reject them. [9]

4.1. Testing process

The testing process must be carefully defined before testing can begin. The process is more important than the actual testing. Correctly defined, the final system will have realistic goals and predictive qualities. Incorrectly done, it will look successful but fail in live trading. Although this chapter discusses backtest, we start with emphasizing the importance of investment philosophy – everything starts with ideas. Backtesting should be the process of validating your investment ideas. If the test results confirm your ideas and investment philosophy then you can have confidence in the trading strategy. If you indiscriminately test all indicator combinations, you will have no idea whether you are finding a good trading strategy or simply overfitting the data. Re-emphasizing the importance of investment philosophy is to underline the importance of backtesting to validate mathematical models and the investment ideas behind the math. Numbers are like people; torture them enough and they will tell you anything (Anonymous - quoted by John Ehlers in Rocket Science for Traders). So before fully kicking-off the testing, you should have the design document in place and fully understand the investment ideas and expectations. The following ideas of deciding value range, value distribution and ranking parameters were adopted from Perry J. Kaufman's book, *New Trading Systems and Methods* (Wiley, 2005).

4.1.1. Decide value range

There are a large number of tests with high frequency data if you have multiple parameters. Due to the vast amount of data, each sample run on minutes-based high frequency data may take much longer than on daily data. If the trading strategy is based on the tick data, the time each run takes is significantly larger than the 1-minute data. See Figure 12, the estimated test time for USDCHF short term momentum trading system.

[9] Definitions of backtest are based on *Quality Money Management* (Kumiega and Van Vliet, 2008)

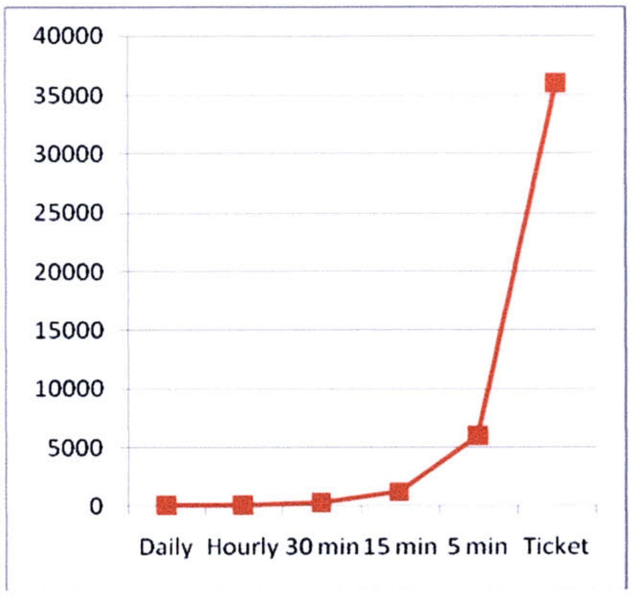


Figure 12: Test time (minutes) for USDCHF on short term trend trading

You must select fewer values to test for each parameter so choose them carefully and wisely based on the statistics from market studies. Tests with many parameters on high frequency data can be extremely time-consuming. In *New Trading Systems and Methods* (Wiley, 2005), Kaufman pointed out that limiting the test is actually an advantage because the value selection process forces you to find the most reasonable value ranges for the trading strategy. "Without these restrictions, the test process gets closer to indiscriminant exploration, and less of a validation process."

4.1.2. Decide value distribution

After deciding the value range, you should also examine the values in the range and decide if you really need them all. For most cases, it is not necessary or practical to test all the values for a valid parameter, considering you have large amounts of intraday data. The values do not need to be spaced evenly apart. [10] For instance: 1, 2, 3, 5, 10, 15, 30, 60 would be a better choice of values than using all numbers between 1 and 60, because, in terms of percentage of change, the

[10] See more details of identifying parameters including deciding value range, value distribution and ranking parameters (next page) from *New Trading Systems and Methods* (Perry J. Kaufman, 2005)

difference between 1 and 2 will be significant, but the difference between 59 and 60 minutes is negligible. Using equal increments will weight this set of tests heavily towards the long end; therefore, the comparison is not very reasonable to some extent. In practice, you can use methods like exponential smoothing to reduce the number of tests. You will achieve significant benefits by limiting the values used in the test.

4.1.3. Rank parameters

Testing one parameter at a time can give you a better understanding of the dynamics of a system. It is hard to understand the dynamics between different parameters when testing a few parameters simultaneously. The most important parameter should be tested first, and a parameter ranking system is needed. This ranking should be based on the trading model logic and the correlations with historical performance. For instance, risk control using the stop-loss, or time-based exit should improve the final model at the later stage of trading model research. However, it is not likely that simply cutting losses will generate net profits, so it does not make sense to rank the stop-loss point as the first parameter tested. Ranking parameters also gives you a good chance to rethink and validate all parameters used by the trading algorithm. When ranking parameters in practice, you can close out some parameters if they turn out to be insignificant statistically. You should capture relevant parameters that impact the trading algorithm but be aware that the more parameters you have, the greater the risk of overfitting.

4.1.4. Decide trading cost

High frequency trading has the advantages of smaller actual loses and more trades. A high frequency trading system that performs well with more trades will give more confidence of future performance. On the negative side, high frequency trading has to fight against slippage and overall trading cost. The trading cost has a significant impact on high frequency trading models and can change the appearance of the multiple-test performance pattern. The trading cost usually includes the spread, slippage and brokers' commission. Before you run backtesting, you should have a reasonable estimation for the trading costs you may encounter during live trading. Unrealistic estimation of trading costs can lead to the failure of live trading.

4.2. In-sample and out-of-sample test

"Performing proper in-sample test and out-of-sample tests is perhaps the most critical step in the development process of a trading system."(Kumiega and Van Vliet, 2008) Once you decide the test size, the parameters and value range, you can start performing the in-sample and out-of-sample tests. In-sample will be run on both train and validation data. Out-of-sample will be allowed on out-of-sample/test data only. "Financial engineers are aware of the extent to which in-sample results may differ from out-of-sample results." (Kumiega and Van Vliet, 2008) When this happens, the trading algorithm must be examined and its parameters and trading rules need to be re-verified. A robust system should be profitable on both in-sample and out-of sample. Always save the test partition and never tune your system based on test data. A backtest result will yield one of three possible outcomes.[11]

- *Profitable in both in-sample and out-of-sample.* In this case, trading models will be accepted and the production will start as soon as possible.

- *Profitable only for in-sample but not out-of-sample.* A three-step approach needs to be taken. First, check the prototype code and make sure there are no programming errors. Next, tune the parameters and re-test the models to see if you can achieve profitability in both in-sample and out-of-sample. After a few rounds of testing and if the model is still profitable in one test and not the other, you reject the model and move to the third step which sends the 'rejected' model back to the strategy developing phase and continues the research.

- *Unprofitable in both in-sample and out-of-sample.* The model will be rejected and for most cases it will be discarded for good, unless the investment/trading idea is very solid and supported by premier research. In this case, you may send the models to the strategy developing phase for continuing research.

[11] See the discussion of three possible outcomes of a backtest in *Quality Money Management* (Kumiega and Van Vliet, 2008)

4.3. Avoid overfitting

4.3.1. Save out-of-sample/test data

As discussed previously, you need to partition the data into separate pieces. You can then build a model using one dataset ("training dataset" or "In-Sample") and evaluate its fit by applying the model to another set ("test dataset" or "Out-of-Sample"). Invariably the out-of-sample performance will be considerably less than the performance generated in the optimization. A tendency for system designers when they see unsatisfactory out-of-sample results is to repeat the optimization process until the out-of-sample results 'become' better. Eventually the out-of-sample data becomes the same as in-sample data and the process gets closer to overfitting. Thus you might alternatively want to use three sets: training dataset (to build the initial model), validation dataset (to guide the progression of the model), and a test dataset to gauge the accuracy of the final model. A method to check whether a good performance is truly generated from the model, or simply just because of overfitting, is that you can test the same model with different currency pairs, or even different asset classes. If the same parameter set works in different markets then you have a reliable and robust system. This seems counter-intuitive because you would expect that each currency pair or each market is different and requires different parameters and values. However, if a high frequency trading system needs different parameters for different currency pairs, it might be an indication of overfitting.

4.3.2. Different time horizons

During a backtest, returns over different periods should be consistent. For instance, a high frequency model with parameters optimized over a 3 year holding period should have a similar return over 6 months, 1 year, 2 year and 5 years. Other measurements such as sharpe ratio, standard deviation and daily maximum drawdown should be consistent in different holding periods. If a high frequency trading model does not work or is not consistent in different time horizons, it is a sign of over-fitting and you should reject the model. Many high frequency models are market neutral and are able to deliver consistent returns during different time horizons.

4.3.3. Different time frames

Like the ideas at different time horizons, a trading system that is built on a 5-minutes bar should work for on different timeframes such as 1-minute and 15-minutes. Returns over different timeframes should be consistent. If a high frequency trading model works on 5-minute bar data but totally failed on 15-minutes bar, a further investigation is needed.

4.3.4. Different Instruments

The consistency of returns on different instruments is a good indication of robustness for a trading system. If trading system is designed and optimized for USDJPY and the same system works for EURUSD without changing any parameter, then you know the system is robust. The characteristic of robustness can help you detect overfitting issues for certain models. If a trading system works well with USDJPY but fails with EURUSD, it raises a red flag and you might have to investigate. This is especially true for high frequency trading systems that are price data based and use technical trading strategies, such as momentum, break-out and resistance/support line.

4.3.5. Different asset classes

Another method of reducing the effect of overfitting is to use more than one market as an out-of-sample test. It is difficult to have the same parameters set in different markets and at the same time overfitting. This is counter-intuitive because you would expect that each market has its own personalities and requires different parameters. However, the high specialization of a trading system (only works for one asset class) usually indicates that the results are from overfitting, not live trading performance. A reliable system should work in most markets.

4.3.6. Student T-test

Among tests that help determine whether the results are significant, the student t-test is the most useful. It helps financial engineers to detect a systematic bias in the data by showing that the mean of the data is significantly different from zero. This information helps us decide whether the results of only a few trades represent a good system, or whether a series of losing trades implies that a system has no value. For a series set of trades produced by backtesting and live trading, the t-test is:

t= average trade results/1 STD of trade results * (number of trades) ^ 0.5

The values of t that are needed to be significant can be found in the T-Distribution table. For instance, if you have 31 trades, the degree of freedom is 30, and the value of T must be greater than 2.423 to achieve 99% confidence in results.

4.3.7. Case studies

The short term trend trading system is built for USDCHF on a 5-minute bar. To test for robustness and overfitting, the two following tests were performed:

- *Test the trading system on different time frames*. This original trading system is built and optimized based on a 5-minute bar. Without changing any parameters, the out-of-sample test was performed for 1-minute and 15-minutes bar data from 2008. See Figure 13. The monthly returns are relatively consistent among all three time frames. The annualized return for 1, 5 and 15 minutes are 1.62%, 4.53%, and 4.22% respectively. The trading system is relatively robust and the chance of over fitting is small.

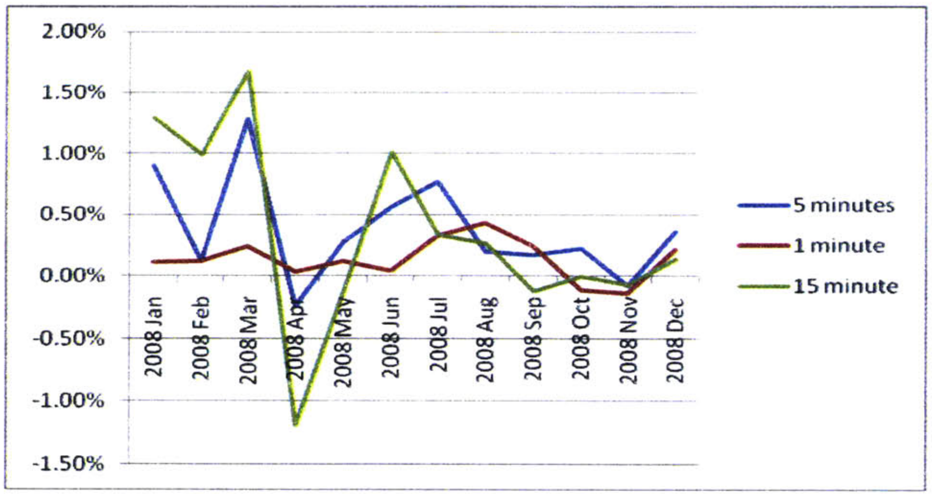


Figure 13: Different time frames for USDJPY on short term trend trading

- *Test on different currency pairs.* The trading system was built for USDCHF. Also performed was an out-of-sample test on the data of 2008 for USDEUR, USDJPY and USDCAD. See Figure 14 for results. The trading system failed on EURUSD and USDJPY. The first impression would be that the trading system has an over-fitting issue because it only 'fits' the currency pair the system was built and optimized upon. However, taking a closer look at the design document and production code, it seems that the problems are related to hard-coded absolute values, such as the historical volatility and stop-loss value.

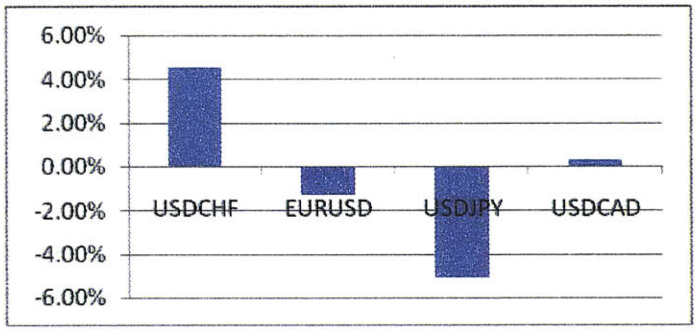


Figure 14: Returns of different currency pairs

When you say "historic volatility < 12 basis points" in the design document, it introduced the 12 basis point into the system as an absolute and hard-coded value. A better approach is to use the relative value so the real value can be calculated against each currency in real time. For instance, instead of saying "historic volatility < 12 basis points", say that "historical volatility < 0.95 * simple moving average of last 40 bars". Thus the value of historical volatility is dynamically calculated based on the simple moving average for each currency pair in run time. The calculations of take-profit and stop-loss values can use the same approach. See Figure 15 that presents out-of-sample test results after eliminating a few hard-coded values and re-optimized based on USDCHF. Although the return of USDCHF is reduced, the returns across different currency pairs are more consistent. EURUSD and USDCAD have a similar return during the basktest without any adjustment of parameters. However USDJPY only has a very few trades during 1 year period in the backtest and this indicates that there is a need for future improvement in the implementation.

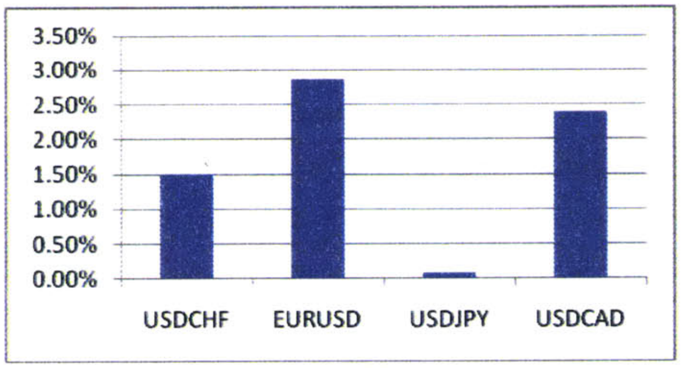


Figure 15: Returns of different currency pairs after eliminating hard-coded values

Based upon the above two tests, I would argue that the implementation of a trading system itself is vital to the robustness of a trading system and minimizes the overfitting. So when we see signs of over fitting, it doesn't necessarily mean the investment/trading ideas are bad. The failure may arise from the implementation itself.

4.4. Trading statistical data

After backtesting, it is important to analyze the statistical data. Not only will it help to measure performance, it will also identify the weakest areas, such as strategies taking too much liquidity and depending on the market conditions. Chapter 6 discusses statistical data and how to measure systems' performance.

5. Optimization

Optimization is the process of using historical data to test the effects of changes in parameters of a trading system to produce the maximum value of the test objective. The test objective can be net profit, sharpe ratio or a robust solution. An important benefit of optimization is that the designer may find parameters and rules that do not work under any circumstances and can be eliminated. Another benefit of optimization is to maintain peak performance in a trading system under the continuously changing market. Although often beneficial, optimization can be one of the most misused techniques. The principal concern with optimization is the tendency to overfit. Overfitting occurs when the optimization program finds the absolute best set of parameters to the training data and the model is twisted to fit the data set being tested. (Kirkpatrick and Dahlquist, 2008) Avoiding overfitting was discussed in Chapter 4. There are many ways to optimize your trading system. The most basic approach is sequential testing that tests different combinations of parameters. The complex techniques may include genetic algorithms and Monte Carlo sampling. For sequential testing, methods including whole sample, out-of-Sample and walk/step forward optimization are discussed in *Technical Analysis* (Charles D. Kirkpatrick, Julie R. Dahlquist. 2008). They are simple and effective. I therefore summarized them in the following discussion.

5.1. Methods of optimization

5.1.1. Whole sample analysis

A whole sample analysis is to run optimizations on the entire sample data. Without out-of-sample data for validation, this approach has a tendency of overfitting. But in some cases whole sample optimization will be useful:

- Use whole sample on one of the currency pairs to train the model, and then test it against other currency pairs.

- After determining the optimal parameter sets, divide the optimization period roughly into tenths to fifteenths and run a test on each period using the optimal parameter set. Then you can analyze the results from the ten to fifteen different test periods to see if the system generated consistent results under all conditions. (Kirkpatrick and Dahlquist, 2008) Consistency is important. If the results are not consistent, the system has a major problem and should be optimized using other approaches. If it still does not work, then you need to reject this trading system.

5.1.2. Out-of-Sample analysis

Out-of-Sample analysis is a commonly used approach that divides the data into two sections, one of 70-80% percent of the data to be used to develop the system called in-sample data, and one of 20-30% called the out-of-sample data. This concept of data partitioning is well known. The 20%-30% out-of-sample data usually are the most recent data. This OOS method optimizes based on the in-sample data and then tests it on the out-of-sample data. If the result of the out-of-sample period is similar to the in-sample performance, the system is considered validated.

A key here is to include bull, bear, sideways movements and a good number of price shocks of various sizes. [12] A tendency for the system designer when they see unsatisfactory out-of-sample results is to repeat the optimization process with different parameter values to make the out-of-sample results 'become' better. Eventually the out-of-sample data becomes the same as in-sample data and the process gets closer to overfitting.

For out-of-sample optimization, you can alternatively divide data into three partitions: train data, validation data and test data. The in-sample test and optimization will never happen on test data (otherwise, it is not out-of-sample).

5.1.3. Step forward analysis

It is close to OOS but moves the test window (in-sample). The test window moves forward until the test reaches the most recent data. For each test window, an optimization on in-sample data is run and an out-of-sample test is performed. The results from all testing windows will be analyzed for profitability and consistency. The acceptable parameter sets are those that have consistent performance across different test windows.

[12] In *New Trading Systems and Methods* (Kaufman, 2005) and *Technical Analysis* (Kirkpatrick and Dahlquist, 2008) , authors of both books pointed out that sample data must cover different market circumstances

6. Systems measurements

When analyzing a trading system, you must look at the system's robustness, the profit, the risk, and its investment capacity, and its contribution to portfolio performance.

6.1. *Robustness of system*

Robustness means how strong and healthy your models are – whether it will be able to perform consistently when the market changes, because the market never exactly repeats the past, which is already introduced into the system design as a basis. The formal definition for system robustness from MIT's engineering system division is "Robustness - demonstrated or promised ability to perform under a variety of circumstances; ability to deliver desired functions in spite of changes in the environment, uses, or internal variations that are either built-in or emergent" [In ESD Terms and Definitions, http://esd.mit.edu/WPS/esd-wp-2002-01.pdf]. Particularly for a high frequency trading system, the system's robustness can be considered as a measurement of counter-overfitting thus the indicators of overfitting addressed earlier serve well as the indicators of a trading system's robustness. A quick check of a system's robustness is to determine whether your high frequency trading system can profit consistently in:

- different time horizons
- different timeframes
- different instruments
- different asset classes/markets

6.2. *Profit and risk measures*

Measuring and comparing the profitability of various trading systems requires a number of performance criteria, in order to evaluate any trading strategy during the test phase and to compare those results to an out-of-sample period or to live trading. The major ways to measure the profitability and risks fall into four categories described in the following sections.

6.2.1. Absolute metrics

There are a number of metrics of performance and risks, and you need to determine measurements based on the nature of trading systems and the requirements of investors:

- *Compounded Total Return.* The total compounded return during the testing period. The return is compounded in order to compare with another trading system.

- *Average Return per Month:* It is usually used to analyze the seasonality of return and the consistency of trading performance. Graphical format is usually used.

- *Annualized rate of return.* It can be used for relating the results of a system against that of a market benchmark.

- *Maximum Drawdown:* The peak-to-trough decline during a specific record period and it is measured from the time a retrenchment begins to when a new high is reached. It gives a rough idea of the minimum capital needed to trade in this market. For high frequency trading systems, you can use *intraday maximum drawdown* due to the fact that most high frequency trading positions are closed in a very short time period. A surprisingly small maximum drawdown is a sign of overfitting or too small a test period. It is safer to take the average maximum drawdown from a range of tests than the one from the best result.

- *Average time to recover:* A large drawdown may be inevitable in live trading; a shorter time to recovery is most desirable.

- *Average net return per trade:* This is a great indicator about how sensitive the high frequency system will be to trading cost.

- *Sharpe Ratio:* It tells us whether a return is due to smart investment decisions or a result of excess risk. It is a common measure of the return versus risk of a portfolio or system, but it has several problems when applied to trading systems. First, it penalizes upside fluctuations as much as downside fluctuations. Second, it does not distinguish between intermittent losses and consecutive losses.

- *Sortino Ratio.* It is similar to Sharpe ratio. But it uses downside deviation for the denominator instead of standard deviation, the use of which doesn't discriminate between up and down volatility.

- *Number of Positive Return Months.* During the test period, the number of months with positive return.

- *Number of Negative Return Months.* During the test period, the number of months with negative return.

- *Smoothness of returns:* Investors always prefer the consistency of return so a smoother equity curve is always more desirable than one with high volatility.

A general rule with these metrics is that one would expect, in live trading, that the return will be half of returns generated in backtesting and the risk (maximum drawdown etc) will be two times the backtesting result.

6.2.2. Trade statistics

Trade statistics are important to understand the reliability and health of the trading systems. They are also very useful pieces of information that can help us with understanding risk and cash management.

- *Number of trades:* This indicates whether your test was statistically reliable. A high frequency trading system generates a larger number of trades so it has a better chance of performing up to expectation.

- *Percentage winning trades*: The more winning trades your trading system has, the better chance of its being profitable in live trading.

- *Average holding period:* All else being equal, a trading system that holds positions shorter than another system is preferable.

- *Averages profit each trade:* A high frequency trading system is extremely sensitive to transaction costs and this figure will tell you how sensitive your trading system will be against the trading cost. If this value is too small, it means that your trading system is very vulnerable to transaction costs.

- *The distribution of different types of orders.* The shares of the total number of trades for Limit orders, Stop Orders and Market Orders respectively. In Figure 16, the distributions of different types of orders are not desirable due to the fact that the majority of orders are stop orders so the system may have very high slippage when the market become volatile and liquidity becomes an issue.

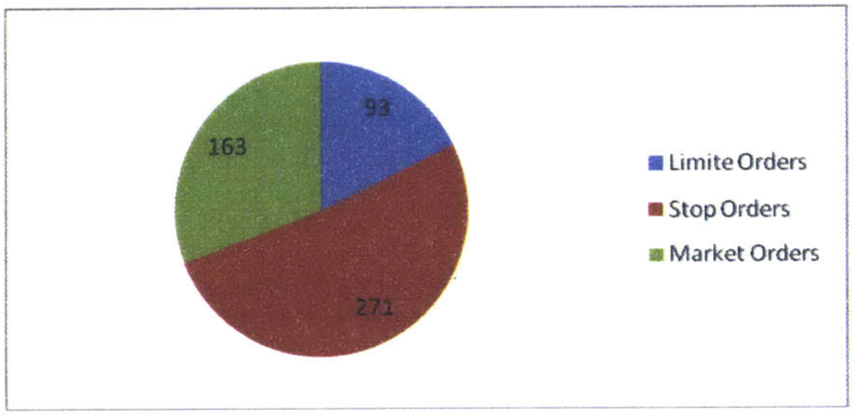


Figure 16: Sample distribution of trading order types

6.2.3. Benchmarking with index

Benchmarking is another way to measure the performance of a trading system or the portfolio. First, it shows to the investors how much you out-perform or under-perform the stock indexes. Second, it tells us the correlation between our trading systems and stock indexes. As Figure 17 shows, the performance of a portfolio of FX high frequency trading systems out-performs the Shanghai Stock Exchange (SSE) and Dow Jones Industrial Average (DJI). The portfolio equity curve is smooth and it is not impacted by the bullish/bearish stock markets.

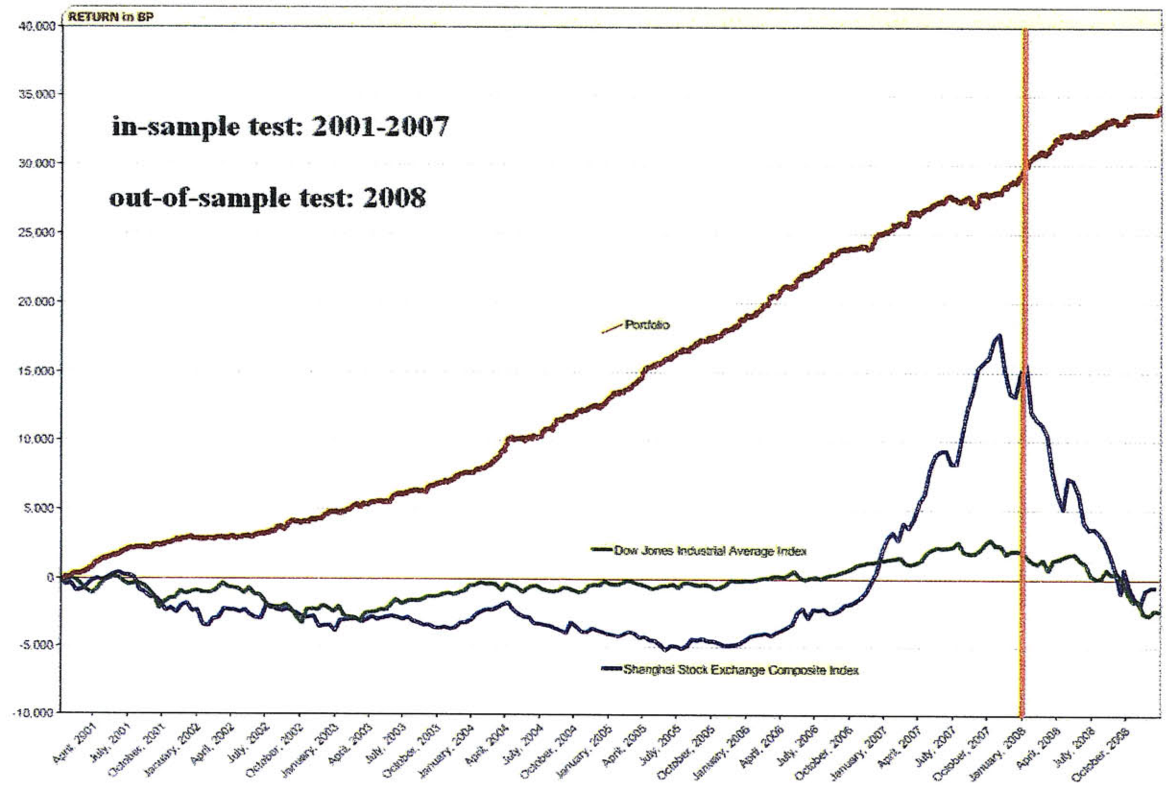


Figure 17: Portfolio performance comparing to stock indexes

6.2.4. Correlations with index

Another important measurement of a high frequency trading system is its correlation with traditional assets classes. A lot of investors would consider high frequency trading products as a diversification of their investment portfolio so it is useful to calculate the return correlation between a high frequency trading system and major stock markets. As figure 18 shows, a portfolio of FX high frequency trading systems has very low correlations with major stock market indexes so it has great diversification potentials for investors.

DJIA	-0.04
S&P 500	-0.02
NIKKEI 225	0.04
Hang Seng	-0.16
DAX	-0.04
FTSE 100	-0.01
Shanghai SE Composite	-0.04

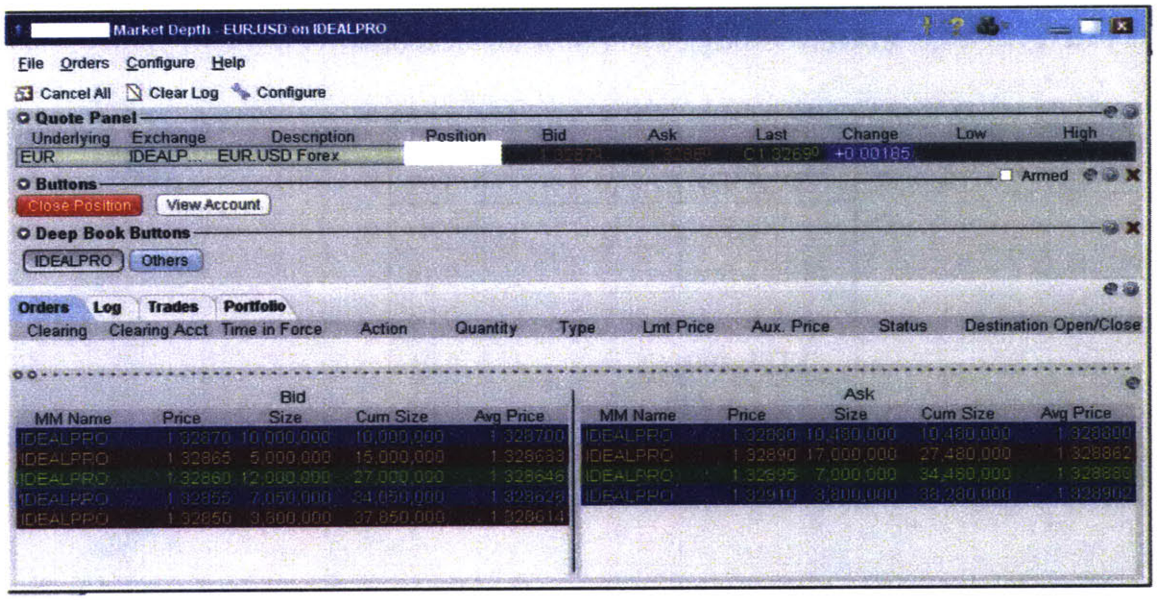
Figure 18: Correlations with stock indexes

6.3. *Investment capacity*

Investors are concerned with the capacity of your trading systems. They want to grasp how much money you can manage. Calculation of investment capacity is not easy for any asset class. There are different systematic approaches to calculating the investment capacity for different markets and different investment products. A simple approach I adopted to calculate investment capacity for high frequency currency trading systems is based on the liquidity your brokers can provide in your trading hours and the commission they ask for. There are three factors in the calculations:

- Market Depth in broker's trading book
- Broker's commissions
- Trading hours

For instance, if a new trading system can only accept 2 pips as the maximum trading cost after the broker's commission and spread, you need to look into your broker's trading book to see how much liquidity you can get in 2 pips during your normal trading hours. Figure 19 is a screenshot taken from Interactive Broker's Trader WorkStation (TWS). There is Euro $37,850,000 as the cumulated tradable size (in Bid) for EURUSD in less than 2 pips. You can average the tradable size in the normal trading hours of your systems so you can get a good estimation of how much you can trade/invest for this particular trading system.


[redacted] Market Depth - EUR.USD on IDEALPRO
File Orders Configure Help
Cancel All Clear Log Configure
Quote Panel
Buttons
Armed
Close Position
View Account
Deep Book Buttons
IDEALPRO
Others
Orders Log Trades Portfolio

Underlying	Exchange	Description	Position	Bid	Ask	Last	Change	Low	High
EUR	IDEALP...	EUR.USD Forex	[redacted]	1.32870	1.32880	C1.32690	+0.00185		

Clearing	Clearing Acct	Time in Force	Action	Quantity	Type	Lmt Price	Aux. Price	Status	Destination Open/Close

Bid MM Name	Bid Price	Bid Size	Bid Cum Size	Bid Avg Price	Ask MM Name	Ask Price	Ask Size	Ask Cum Size	Ask Avg Price
IDEALPRO	1.32870	10,000,000	10,000,000	1.328700	IDEALPRO	1.32880	10,480,000	10,480,000	1.328800
IDEALPRO	1.32865	5,000,000	15,000,000	1.328683	IDEALPRO	1.32890	17,000,000	27,480,000	1.328862
IDEALPRO	1.32860	12,000,000	27,000,000	1.328646	IDEALPRO	1.32895	7,000,000	34,480,000	1.328880
IDEALPRO	1.32855	7,050,000	34,050,000	1.328628	IDEALPRO	1.32910	3,800,000	38,280,000	1.328902
IDEALPRO	1.32850	3,800,000	37,850,000	1.328614					

Figure 19: Market depth for EURUSD

7. Conclusion

The goal of this research was to demonstrate how the application of academic research and the native processes and well-established, coherent design principles of an IT engineering product framework could improve the design, development and reliability of high frequency financial trading systems. By nature, computation-based high frequency trading systems are highly reliant on software engineering systems. Developing a successful high frequency trading system can increase the competitive advantage and profitability of investment firms. Being able to convert academic research results and mathematical equations into a profitable trading system quickly is critical. From a systems and a business perspective, the ability to implement high frequency trading algorithms in a real-world trading environment efficiently and effectively is a key competitive advantage. Investment firms that succeed acquire great leverage in the new world of global, high-speed trading. A major point of emphasis in my research is to point out and demonstrate how the application of a coherent system design framework and rigorous implementation processes, such as backtesting and optimization, is the most important part of high frequency trading system development.

My research used two approaches (SysML and OPM) of model-based systems engineering (MBSE) to architect and create development process models for a high frequency trading system. My thesis shows that well-defined system models and processes are fundamental to ensuring repeatable success and consistent performance. Because a high frequency trading system is inherently complex and requires precise calculation and fast execution, MBSE offers significant potential benefits to facilitate a high quality design, improve productivity and reduce the development risk. Arguably, without any system modeling, exceptional people may still build a good trading system. However, making success highly dependent on extraordinary individuals is risky, unsustainable and does not scale. Although there is no one standard system design framework and set of processes, as is true for all engineering disciplines, the approach selected and the tools chosen depend on the nature of the problem, the skill of the engineers, and the development time and budget. To maximize the chance of success and minimize the risk of failure, this thesis shows that building a high frequency trading system requires the application of a proven design framework and process management principles.

High frequency trading systems deal with large amounts of data and are extremely calculation-intensive. You must have a strategic plan to backtest effectively and efficiently. Without using appropriate backtest techniques, the differences in system performance can be enormous. My research also points out that backtesting and optimization is more important than the trading algorithms used in developing a system. You can build a good trading system with a naive model if you backtest and optimize it correctly. With the right backtesting approaches and techniques, you can prove that your models will not work if they are incapable of avoiding the loss of capital in live trading. Similarly, the misuse of backtesting and optimization can lead to trading failures, even if the underlying trading algorithms and rules are sound.

Lastly, my research concludes that flexible, continuous system improvement is also fundamental to success. The design framework, development processes and backtesting techniques employed in system development need to be constantly reviewed and improved. Rigid design principles and processes do not optimize chances of success. The inherent nature of high frequency trading requires systematic innovation, development and continuous improvement to guarantee the quality of design and performance of processes of high frequency trading systems.

Appendix I: Trading System Development Process

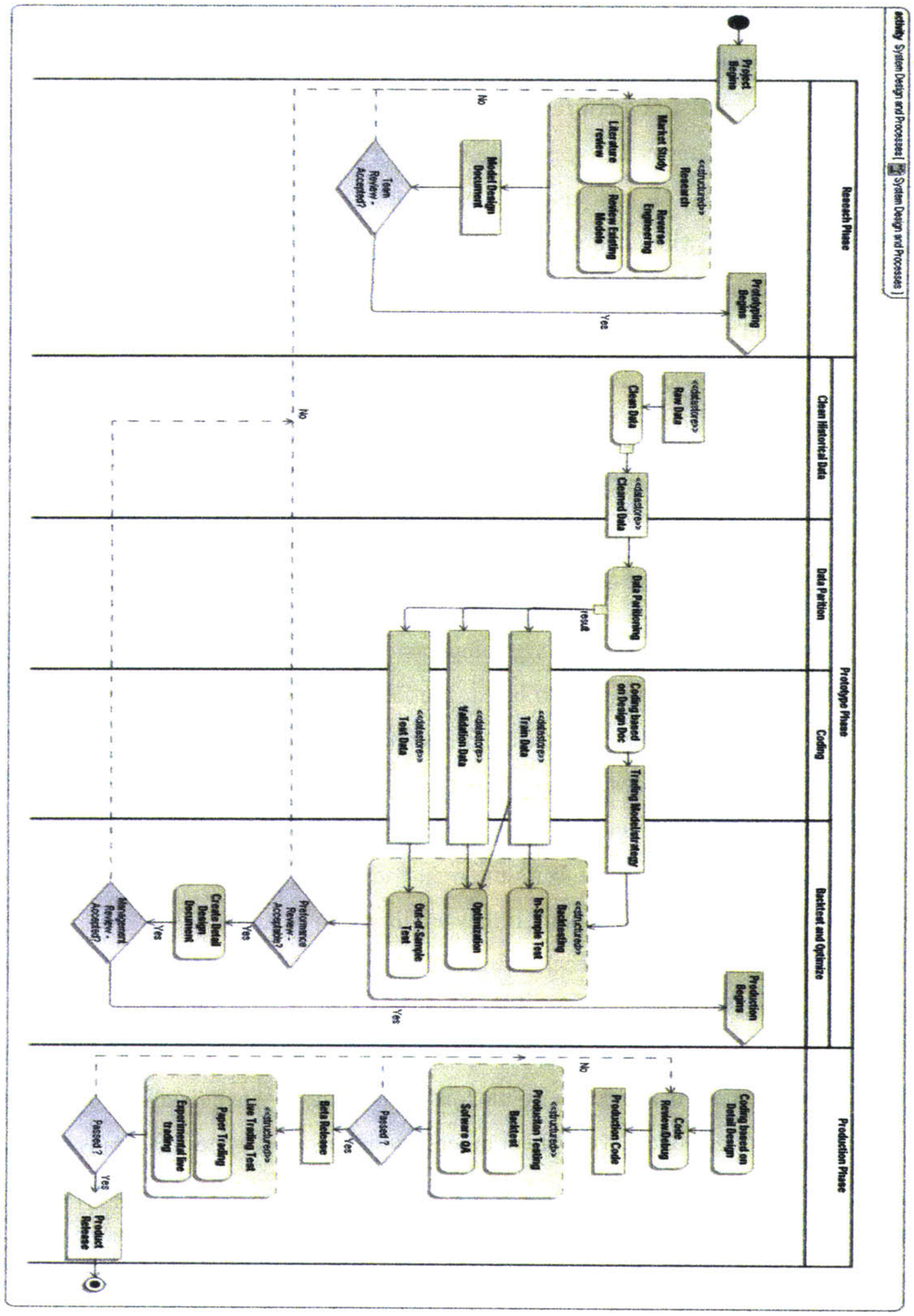


Figure 20: Trading system development framework and process (SysML)

Appendix II: Trading System Modeling with Object-Process Methodology

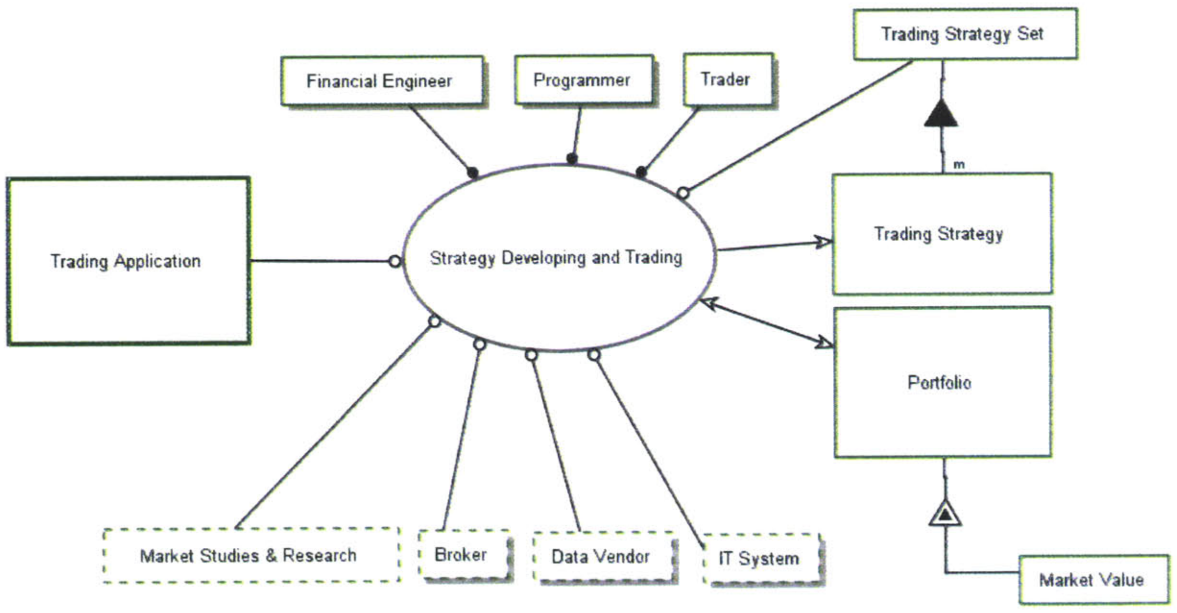


Figure 21: Trading system development framework and process (OPM)

Figure 21 to Figure 28 represent trading system models generated in Object-Process Methodology (OPM). This first diagram is the highest level diagram and the following diagrams are created by zooming in or unfolding on this diagram, each of the following diagrams represents the sub-systems and their internal process. This highest level diagram represents the primary process and components (objects) in which it interacts. The system does two things. First, it creates and modifies Trading Strategies, which is used within the system. Also, it is possible that the trading strategies/models developed by his or her system could be exported into another system. The Trading process also alters the Portfolio object, which is a representation of all the positions held by those using the trading system. The entire purpose of the system is to alter the Portfolio over time, such that its Market Value (a characteristic exhibited by the Portfolio) is hopefully being increased over time.

In order to function, Trading also requires IT Systems, Data Providers, and Brokers as instruments. These are all considered external to the system, since they typically exist and the people implementing this system do not have control of them. The IT System is managed by the IT Staff. This relationship is not represented with an explicit process, because it is not of much importance when describing this system. Financial engineer, programmer and trader also are involved in the Trading strategies and live trading process.

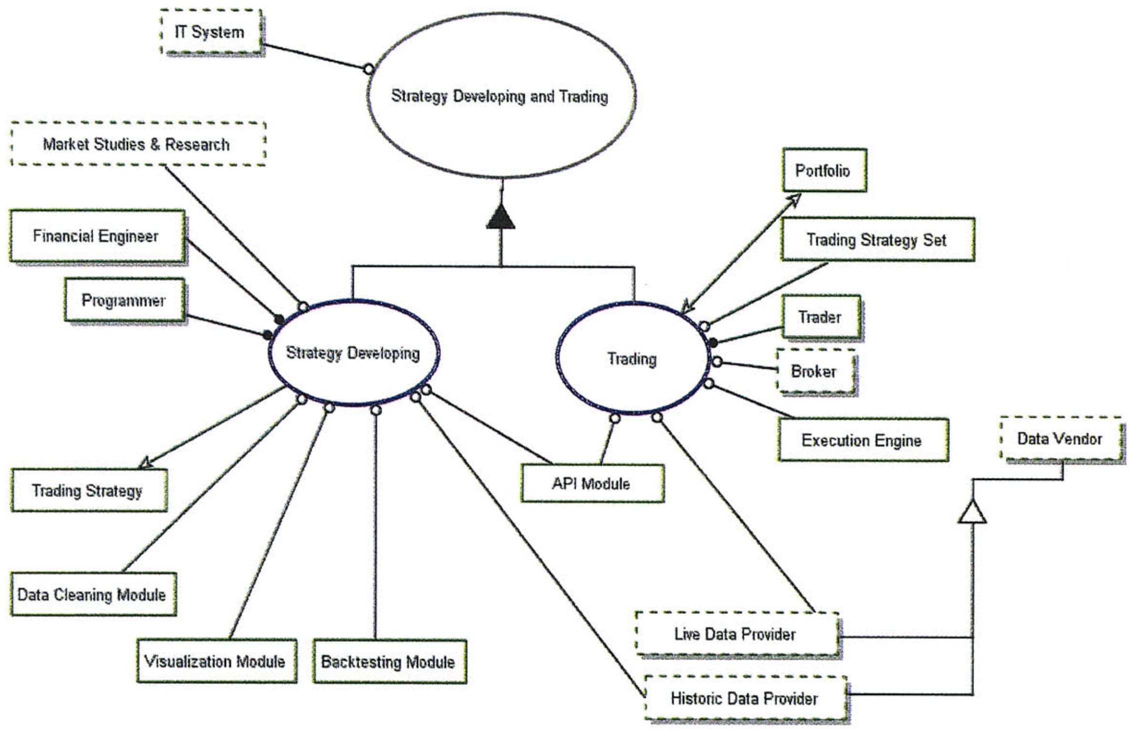


Figure 22: Trading system development framework and process – Unfolded (OPM)

This diagram displays a first level decomposition of the trading system. The two major functions of such a trading system are separated. One part focuses on strategies and the other is for live trading.

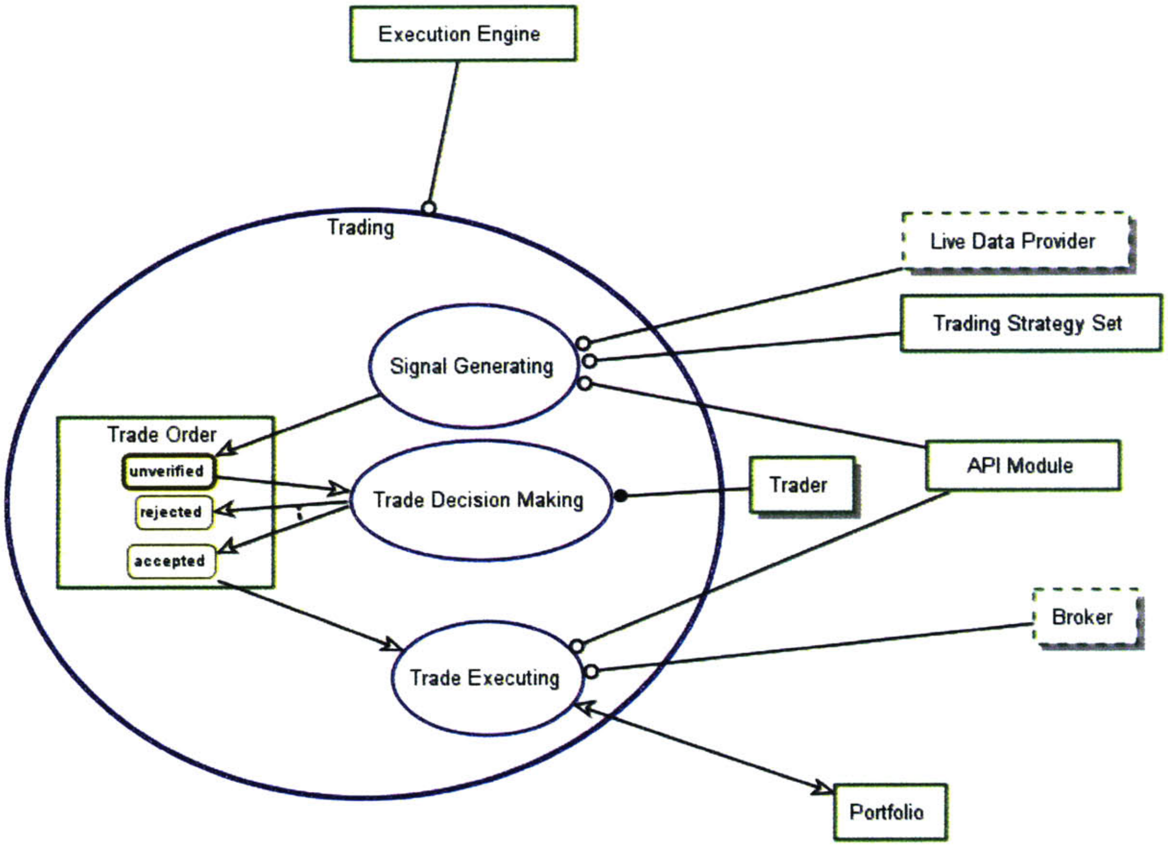


Figure 23: Trading diagram - zoomed-in (OPM)

This trading diagram is a zoom-in from figure 23 which represents the live trading process. And the next figure (Figure 24) is another zoom represents the strategy developing process.

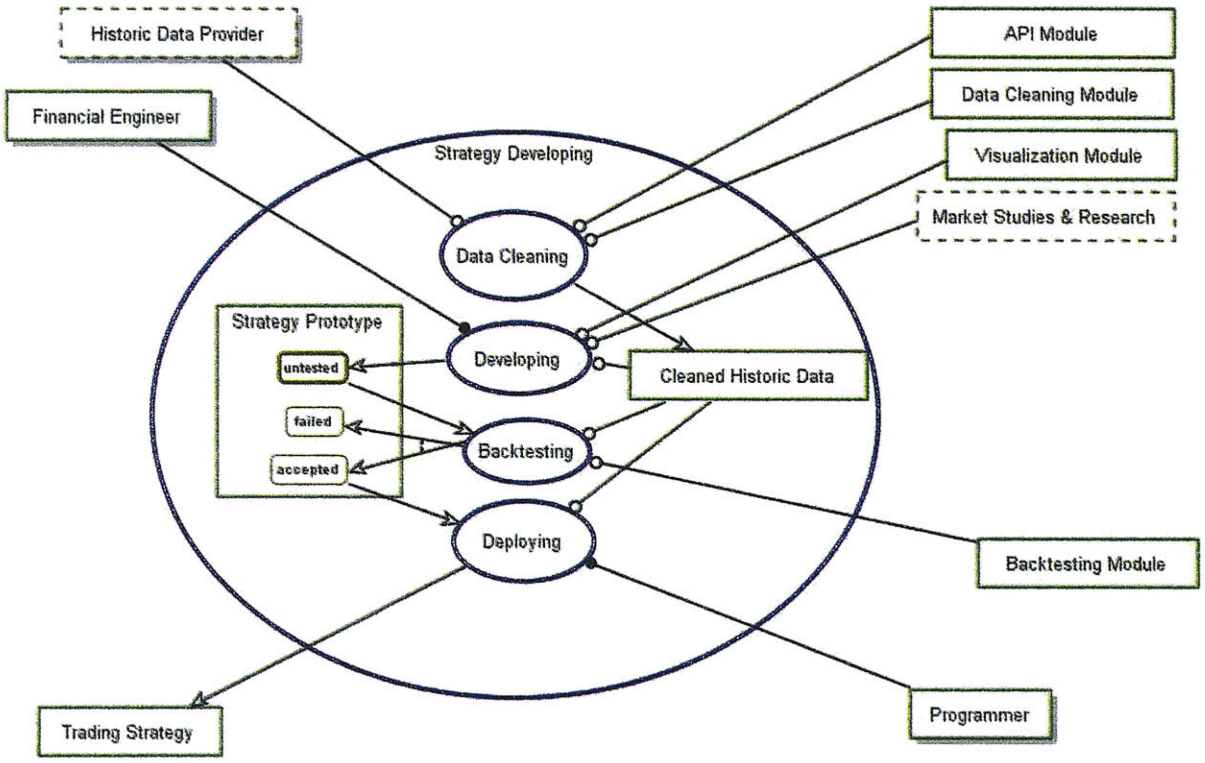


Figure 24: Trading strategies developing diagram – Zoomed-in (OPM)

This diagram is a zoom-in of strategy developing from the upper level diagram. It is composed of four sub-processes, including Data Cleaning, Strategy Developing, Backtesting, and Executing. Typically, Executing is constantly acting, while the other three processes are running in sequence to develop trading strategies.

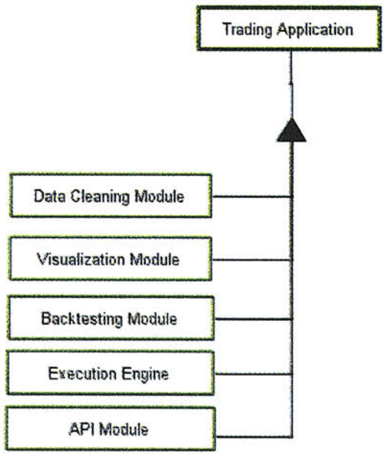


Figure 25: Trading application – Unfolded (OPM)

In addition, the Proprietary software is decomposed into its own objects, which include API Module, Data Cleaning Module, Visualization Module, Execution Engine, and Backtesting Module. The API Module exists simply to connect the Data Providers to the Trading System, which we did not believe warranted the introduction of a specific process. Therefore, it is connected directly to the Data Provider using a structural link. The other four objects each are directly related to sub-processes of Trading. This is intentional – since we are creating software it makes the most sense to organize the application around the primary processes of the application to facilitate development.

The Data Providers object can be specialized into two other objects, Historic Data Providers and Live Data Providers. Each are used as instruments for different sub-processes of Trading.

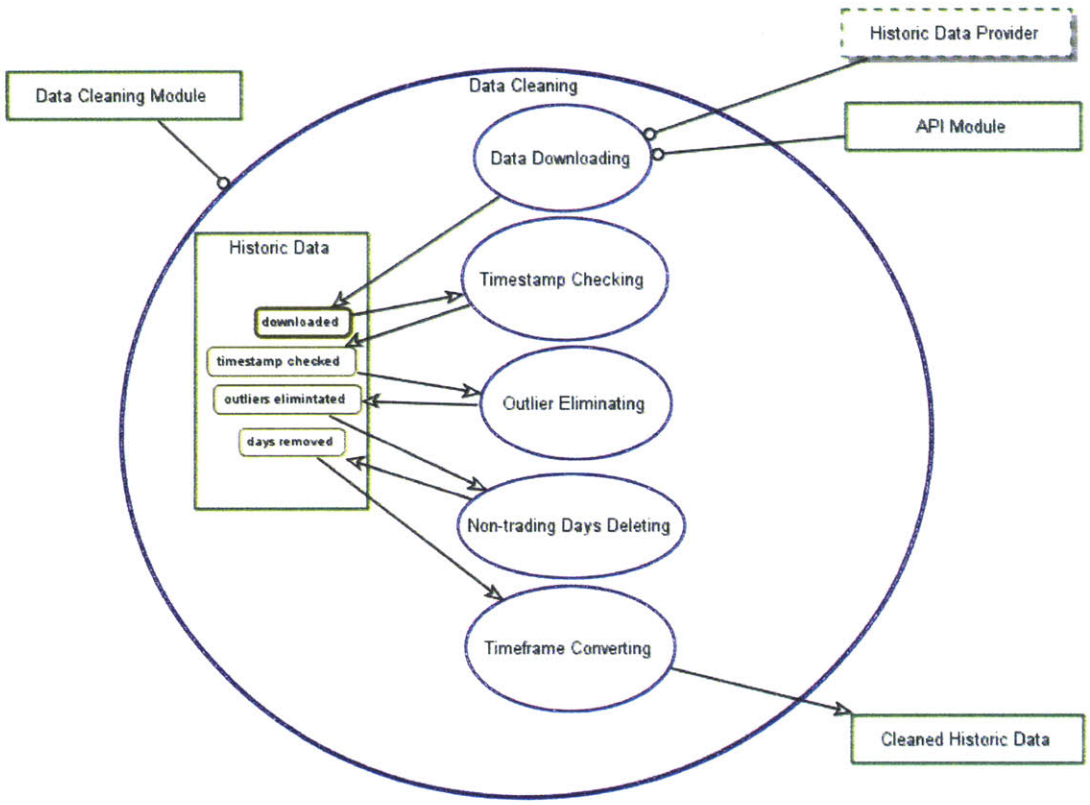


Figure 26: Data cleaning – Zoomed-in (OPM)

This is a zoomed-in OPM diagram for data cleaning that contains sub-processes we have discussed in early chapters based on SysML.

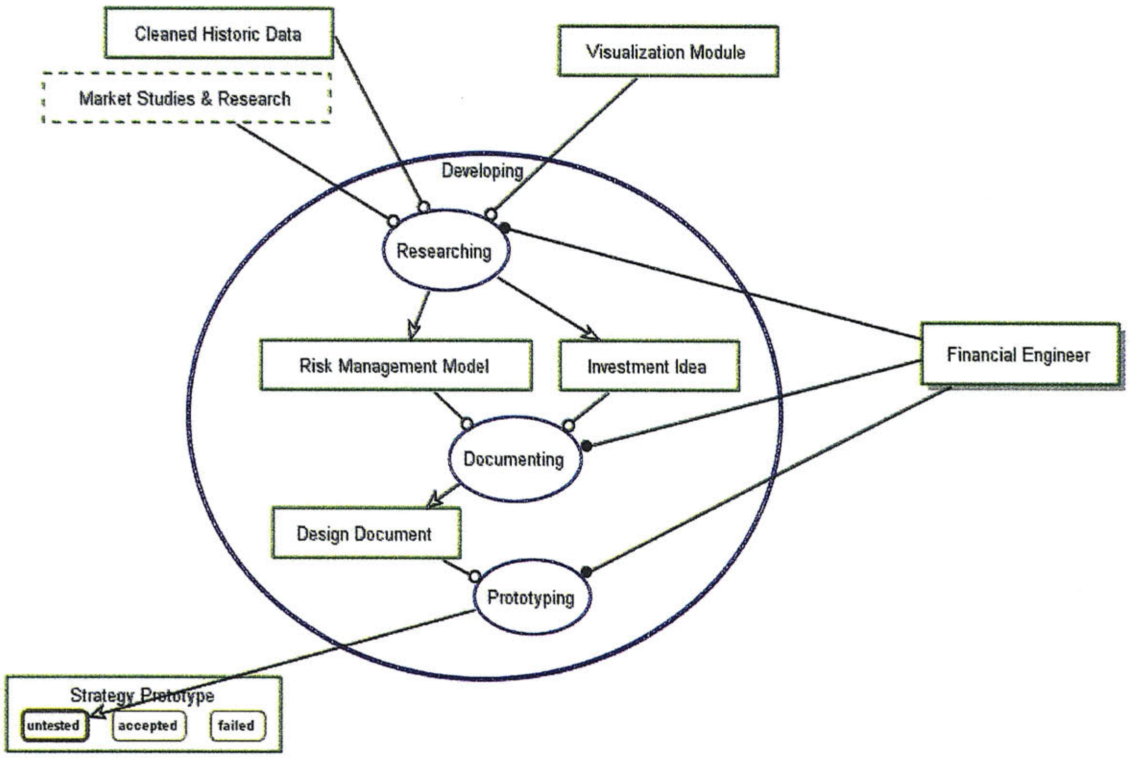


Figure 27: Trading strategies/models developing – Zoomed-in (OPM)

This diagram shows the process of how financial engineers generate investment ideas and risk management models based on research. Then financial engineers create a design document that will be used by prototyping. This process has been discussed in chapter 3.

The next two diagrams are Backtesting and Production, respectively. The processes themselves are explained in chapters 3 and 4 and these two diagrams are presentations of the same processes but in OPM format.

As discussed earlier, both SysML and OPM are powerful and capable. Choosing between them depends on our background, the nature of the projects and your personal thinking process preference.

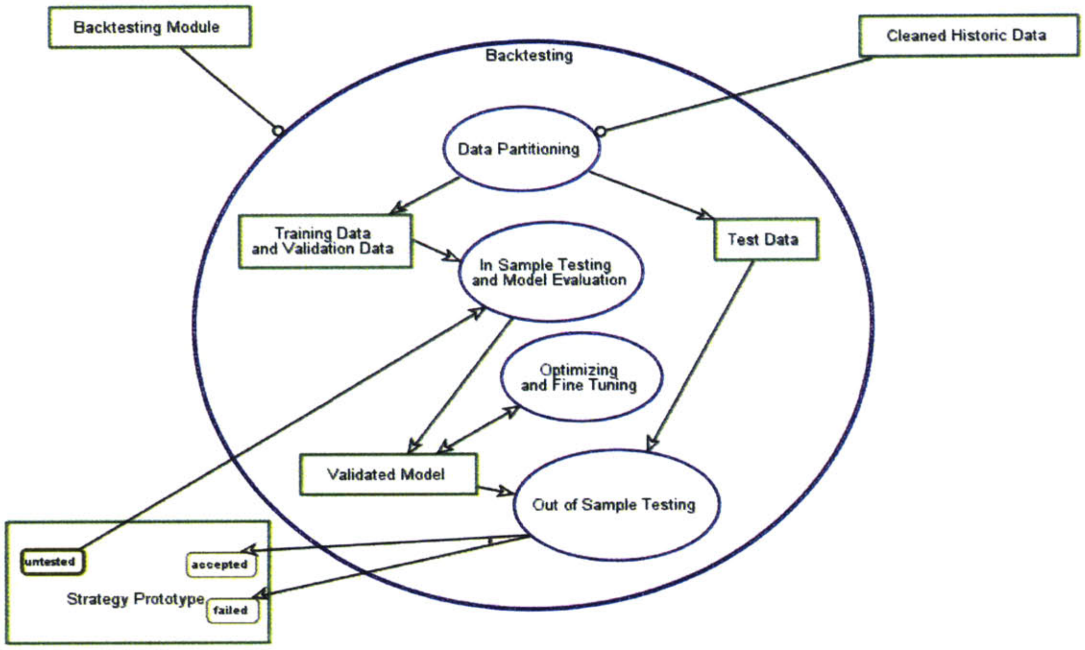


Figure 28: Backtesting process – Zoomed-in (OPM)

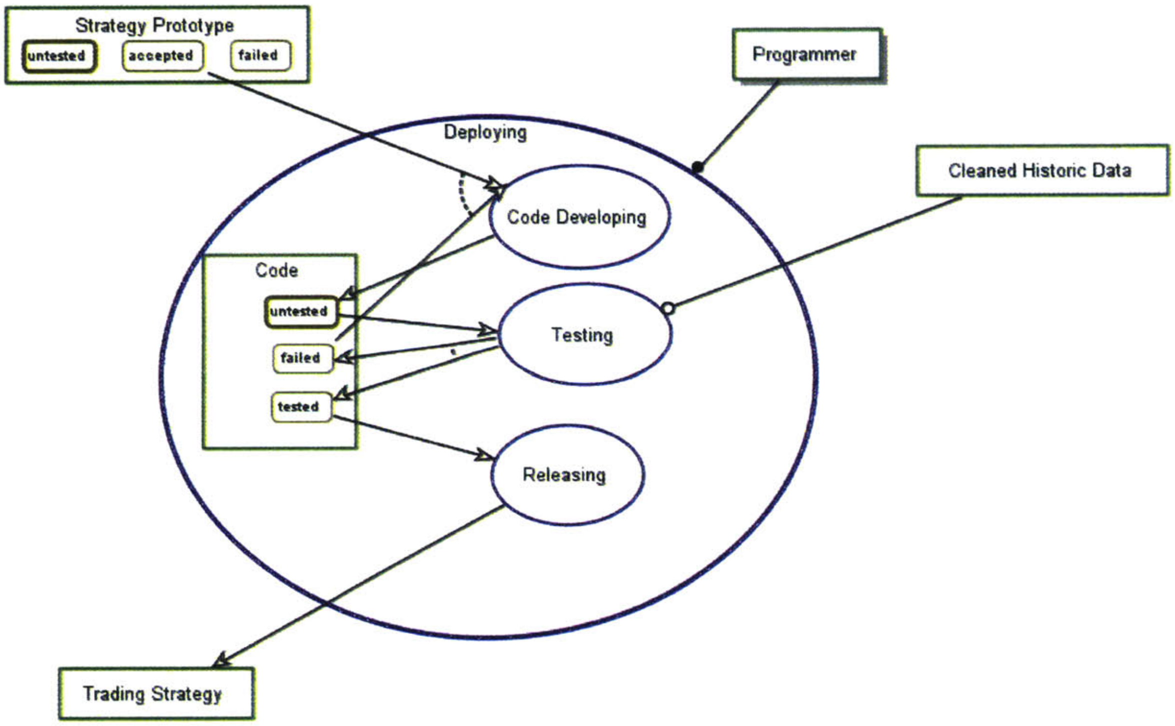


Figure 29: Production/Deploy process – Zoomed-in (OPM)

Bibliography

Michel M. Dacorogna, Ramazan Gençay, Ulrich Müller, Richard B. Olsen, Olivier V. Pictet. *An introduction to High-Frequency Finance.* Academic Pres. 2001

Perry J. Kaufman. *New Trading Systems and Methods*, Wiley. 2005

Andrew Kumiega, Benjamin Van Vliet. *Quality Money Management: Process Engineering and Best Practices for Systematic Trading and Investment.* Academic Press. 2008

Charles D. Kirkpatrick, Julie R. Dahlquist. *Technical Analysis: The Complete Resource for Financial Market Technicians.* FT Press. 2008.

Lawrence Bernstein, C. M. Yuhas. *Trustworthy Systems Through Quantitative Software* Engineering. Wiley-IEEE Computer Society Press. 2005

John R. Wolberg. *Expert Trading Systems: Modeling Financial Markets with Kernel Regression.* Wiley. 2000.

Tushar Chande. *Beyond Technical Analysis: How to Develop and Implement a Winning Trading System, 2nd Edition.* Wiley. 2001

Bingcheng Yan, Eric Zivot. "*Analysis of High-Frequency Financial Data with S-Plus*" Department of Economics, University of Washington. 2003

Galit Shmueli, Nitin R. Patel, Peter C. Bruce. *Data Mining for Business Intelligence: Concepts, Techniques, and Applications in Microsoft Office Excel with XLMiner.* Wiley-Interscience 2006

Sanford Friedenthal, Alan Moore, Rick Steiner. *A Practical Guide to SysML: The Systems Modeling Language.* Morgan Kaufmann . 2008

Dov Dori , *Object-Process Methodology*. Springer. 2002

Edited by Christian Dunis and Bin Zhou. *Nonlinear modelling of high frequency financial time series*. Wiley. 1998.

Edited by Pierre Lequeux. *Financial markets tick by tick: insights in financial markets microstructure*. Wiley. 1999.

Thomas Stridsman, *Trading systems that work: building and evaluating effective trading systems*. McGraw Hill. 2001.

John Kicklighter. "*Top 5 Most Market Moving Indicators for the US Dollar*". DailyFx.com. 2008

Cory Mitchell. "*Trading The Non-Farm Payroll Report*". Investopedia.com. 2009

Kathy Lien. "*Trading On News Releases*". Investopedia.com. 2008

Free Online Forex Trading Course. FOREXHIT.com
http://www.forexhit.com/learn-forex/free-online-forex-trading-course.html#chart