Imperial College London
Department of Computing

Specification and Performance Optimisation of Real-time Trading Strategies for Betting Exchange Platforms

Polyvios Tsirimpas

Submitted in part fulfilment of the requirements for the degree of
Doctor of Philosophy in Computing of Imperial College London
and the Diploma of Imperial College, March 2013

Abstract

Since their introduction in June 2000, betting exchanges have revolutionised the nature and practice of betting. Betting exchange markets share some similarities with financial markets in terms of their operation. However, in stark contrast to financial markets, there are very few quantitative analysis tools available to support the development of automated betting exchange trading strategies.

This thesis confronts challenges related to the generic specification, back-testing, optimisation and execution of parameterised automated trading strategies for betting exchange markets, and presents a related framework called SPORTSBET. The framework is built on an open-source event-driven platform called URBI, which, to date, has been mainly used to develop applications in the domains of robotics and artificial intelligence. SPORTSBET consists of three main components, each of which addresses a hitherto-unmet research challenge. The first is UBEL, a novel generic betting strategy specification language based on the event-driven scripting language of URBI, which can be used to specify parameterised betting strategies for markets related to a wide range of sports. The second is a complex event processor which is capable of synchronising multiple data streams and either replaying them on an historical basis with dynamic market re-construction, in order to quantify strategy performance, or executing them in real time with either real or virtual capital. The final component is an optimisation platform whereby strategy parameters are automatically refined using a stochastic search heuristic in order to improve strategy performance. Explicitly, the optimisation process involves stochastic initialisation, intermediate stochastic selection and acceptance of the candidate solution.

To demonstrate the applicability and effectiveness of SPORTSBET, case studies are presented for betting strategies for a range of sports. As illustrated in the case studies, the SPORTSBET optimisation platform implements Walk-Forward Analysis for the robust parameterisation of betting exchange trading strategies without overfitting. Nonetheless, the outcomes should be carefully interpreted, while numerous tests of a strategy are recommended.

Acknowledgements

I would like to thank the following people, without whom this thesis would never have been possible:

- My supervisor, Dr William Knottenbelt, for his patience, inspiration, guidance and constant enthusiasm throughout my research.
- My family, for their love, support and encouragement.
- Rocio Alvarez for her constant support during my research.
- Tom Freeman and Tom Large from Betfair for their support throughout this research.
- Betfair for providing me with the funding to do my PhD.

In most betting shops you will see three windows marked “Bet Here”, but only one window with the legend “Pay Out”.

Jeffrey Bernard (1932-1997)

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Ever since ancient times, humans have been instinctually fascinated by chance and ways to predict the future [100]. Building on this fascination, gambling – that is, the wagering of money on events with uncertain outcomes in the hope of winning some sort of prize (usually financial) – has naturally grown over the ages into a thriving global industry. Indeed, the UK's online gambling sector was estimated to be worth more than £1.7bn in 2012 (see Figure 1.1).

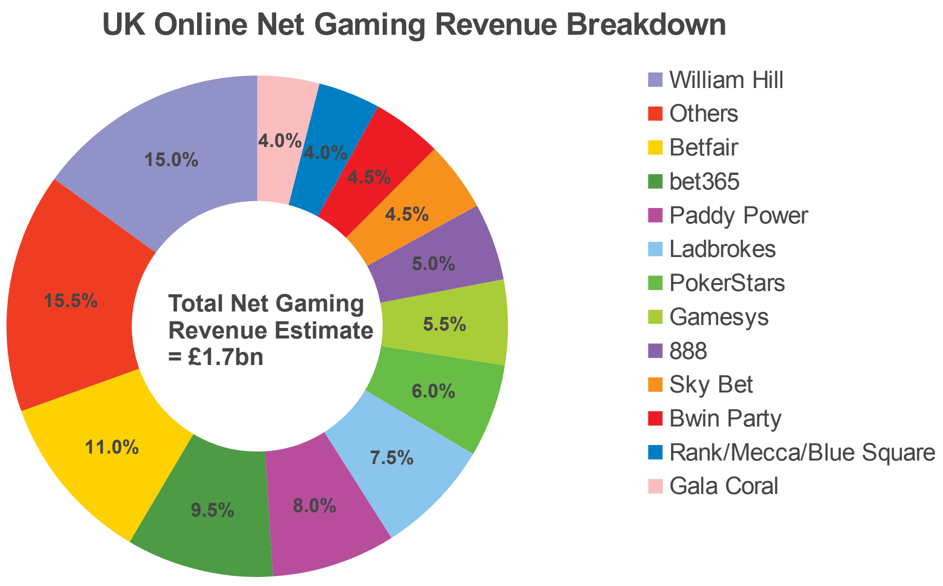


Figure 1.1: UK online net gaming revenue in 2012 [48]

A significant subset of modern gambling industry revolves around sporting events, thus providing customers not only with amusement and entertainment but also with the opportunity to participate in a cognitive challenge based on skill and expert knowledge. Mathematics and statistics have long provided the fundamentals supporting this industry, but Information Technology (IT) has transformed almost all aspects of it. For example, the speed of internet-facilitated information dissemination and rapid rises in the power of computation have seen the rise of "in-play" betting. This allows wagers to be placed at almost any time during the course of an event. Equally, the number of available markets related to each event have multiplied.

IT has most especially transformed the way bets are offered and taken up. Traditionally, bets have been offered by bookmakers (usually companies), and taken up by their customers (usually individuals). Bookmakers have aimed to guarantee themselves a profit by offering bets on any given event outcome at odds which are lower than the (unknown) true odds of the event outcome. The advent of the betting exchange [106, 70] in June 2000 offered a different business model, whereby customers were able to offer bets directly to each other, or take up bets offered by other customers, in a peer to peer fashion. Customers are attracted to betting exchanges by the promise of better prices, while the betting exchange is guaranteed to make a small riskless profit by taking a commission from the net profits of the winner.

Betting exchange markets have increasingly been attracting researchers' interest in recent years as the levels of market activity have begun to approach that of more traditional financial markets. Their growing economic importance becomes obvious from the fact that the Betfair exchange processes more than seven million transactions every day – more than all European stock exchanges combined[1] and also from the recent emergence of firms specialising in quantitative trading of sports betting markets[2]. Moreover they have received much attention in the academic literature due to their similarities to financial markets and their use in studying market efficiency. Thaler and Ziemba [110] were amongst the first to argue that betting markets may be better suited than financial markets when testing for efficiency. The main advantage is that bets have a well defined period of life at the end of which their value becomes certain.

[1]Source: http://corporate.betfair.com/about-us/key-facts.aspx

[2]See e.g. http://acmltd.com and http://www.gambitresearch.com.

This makes their pricing and the subsequent testing of efficiency far less complicated. Direct similarities among specific financial assets and bets have also been discussed in the literature. Ruhm [98] explained how positions in financial options can be viewed as simple bets, whereas, Vecer et al. [118] in their examination of the 2006 FIFA Soccer World Cup compared betting contracts with credit derivatives.

In financial markets the goal of many traders is to buy and sell securities with the objective of maximising profit and minimising risk of loss. Similarly, in a betting exchange market participants are seeking to discover rules which deal with defining the optimal time, price and volume at which to buy and sell bets. Many former financial traders are operating on betting exchange markets and some of them use trading techniques and strategies with origins in the financial trading sector. Trading in the financial stock markets is actually a form of gambling (e.g. when buying a stock, you are taking a gamble that the price will increase in the future, so that later the stock can be sold with profits). Indeed, commentators such as Feustel [42] have gone so far as to term the activity "stock betting".

Despite that betting exchange markets share many of the properties of financial markets there are some important differences. Betting exchange markets have a well defined period of life contrary to most financial stock markets where the life of a market is not known (an obvious exception being certain options and futures markets). Each sport has many different rules and each sporting event can have many different markets, while in financial world there are less rules and more standardised markets. Moreover, in the sports betting and betting exchange world, external events (e.g. a point in a match) tend to be more frequent and standardised compared with external events in the financial world (which may be marked by an occasional news release for example).

In both cases, potential trading strategies may be evaluated using a process called back-testing. Back-testing is a specific type of testing which uses historical or synthetic (controlled) data and aims to calculate how a strategy would have performed if it had actually been applied in the past. This requires the back-test to replicate the market conditions of the time in question in order to get an accurate result. While back-testing does not tell whether a strategy will work in the future, its primary benefit lies in understanding the vulnerabilities of a strategy as it encounters real-world conditions. This enables the designers of a strategy to "learn from their mistakes" without actually having to make them with real money [30, 104].

The research process for a quantitative trading strategy, from conceptual design to actual execution, can be very time-consuming, requiring months or more depending on the strategy complexity. The back-testing process usually takes the longest time. There are numerous details to include in back-testing code. To name a few: data cleaning and preparation, mathematical algorithms, market simulation, execution, parameter calibration, sensitivity analysis and debugging. There are many financial tools allowing traders and researchers to form their strategies and test them against multiple data sources[3]. Yet to our knowledge there is no quantitative trading research platform publicly available for sports betting exchange trading strategies that alleviates traders from coding up those "infrastructural" components.

Many traders specialise in in-play trading, where fast access to streams of information (especially concerning in-game events) is crucial. Any sporting event broadcast live on TV is transmitted with a few seconds delay (3–10 seconds) and therefore bettors with a faster transmission, or those present at the game, are able to trade on an informational advantage. Despite that fact that trading/gambling on tennis matches at the court side is not officially allowed by the ATP, there are some recent cases where bettors have been spotted in tennis matches sitting court-side engaged in trading on portable devices[4]. Another (illegal) way for traders to attempt to gain an advantage is to exploit inside information about the likely development or outcome of a match. This emerged as a major issue following the so-called "Davydenko scandal"[5]. This scandal relates to a Poland Open tennis match played in Sopot in 2007 between Nikolay Davydenko and Martin Arguello. The match was marked by bizarre price movements (such as the odds on Davydenko winning increasing even when he won the first set against his low-ranked opponent with ease), and sparked an ATP investigation into match fixing, leading

[3]See e.g. `http://www.ninjatrader.com/`, `http://www.esignal.com/`, `http://www.tradestation.com/`, `http://www.optionsxpress.com/` and `https://www.tradeking.com/`

[4]`http://eastbournetennis.com/2012/06/tennis-tours-set-to-net-courtside-cheats/`

[5]`http://betting.betfair.com/tennis/general/say-it-aint-so-the-shadow-over-090807.html`, `http://news.bbc.co.uk/sport2/hi/tennis/6928635.stm`, `http://www.thetennisspace.com/the-inside-story-of-the-davydenko-controversy/`

Betfair to take the unprecedented step of voiding all bets placed on the match outcome.

Automated trading systems are dominant in financial markets. However, these systems do not fit the requirements of betting exchange markets. Ordinary programmers, researchers and traders are having a difficult time to construct or use these systems effectively.

We suggest that a generic formal language for describing betting exchange strategies and an environment that allows for their live execution and/or historical quantitative evaluation in addition with their parameter(s) optimisation are essential for rapid advancement in this field. To this end, we have developed a framework called SPORTSBET (*Specification and Performance Optimisation of Real-time Trading Strategies for Betting Exchange plaTforms*). After a trader has coded up a strategy and chosen a parameter set, there is a whole suite of automated analysis that can be followed in order to evaluate the strategy. For instance, the trader can see how the strategy performs for historical data as well as with real time data. Then the trader can construct the profit and loss distribution, do sensitivity analysis of parameters and compute many performance statistics. All of these are very time consuming and CPU-intensive tasks. SPORTSBET simplifies the whole process, and even allows for the subsequent real time execution of optimised strategies with real capital.

1.2 Aims and Objectives

This research aims to develop a virtual laboratory for the definition, development, testing and automated improvement of betting exchange trading strategies.

This high level task entails several supporting objectives:

- Devise a well-defined betting strategy programming language that users can use to specify strategies in a rigorous manner. The language must be accessible to bettors and researchers, yet powerful and generic enough to support a wide range of sports related to betting exchange markets.

- Develop a complex event processor (including strategy execution and market simulation engine) capable of synchronising multiple real-time data streams (market data, in-game events, etc.) and replaying them on an historical basis (with dynamic market reconstruction).
- Build a database infrastructure that will provide a comprehensive, constantly-updated source of sporting events data.
- Formulate appropriate metrics for quantifying strategy performance.
- Construct an optimisation platform that can be used to find (near-)optimal parameterisations of the input strategy.

1.3 Contributions

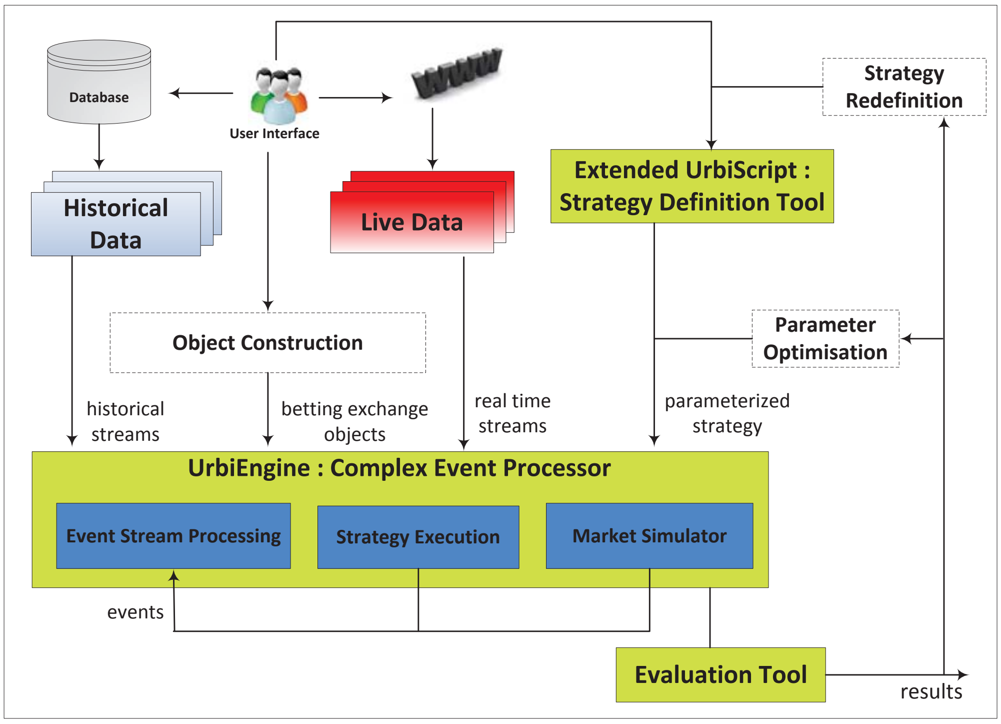


Figure 1.2: The high level architecture of SPORTSBET

This thesis confronts challenges related to the generic specification, back-testing, optimisation and execution of parameterised trading strategies for betting exchange markets, and presents a related framework called SPORTSBET (cf. Figure 1.2). The work presented is innovative and expands the applicability, capacity and specification power of prior work in the research areas of finance, betting and artificial intelligence.

As discussed in the previous section, it is essential to provide users with a means of creating generic parameterised betting strategies. To this end, we have created the Universal Betting Exchange Language (UBEL) which is an extension of the UrbiScript language [108] (a concurrent programming language for robotics). UBEL is the first domain-specific language for betting; it simplifies the process of writing betting exchange market trading strategies and allows them to be expressed more clearly comparing with general-purpose programming languages such as Java and C++. The language is generic enough to support a wide range of sports related to betting exchange markets and naturally supports event-based programming and concurrency.

To evaluate a strategy using historical data accurately, it is necessary that the market simulator replicates the conditions of betting exchange markets as closely as possible. To do that we augmented URBI's scheduler (called UrbiEngine) with processes which model the operation of betting exchange markets and we have constructed a scheduler capable of synchronising multiple data streams and replaying them on an historical basis with dynamic market re-construction. This complex event processor supports the parallel simulation and/or live monitoring of many different sporting events and their associated markets and can place or cancel many bets in parallel across different markets. A bet can be matched, partially matched or stay unmatched.

SPORTSBET can automatically record market prices, events data and corresponding bet details from betting exchange platforms in its databases. This historical data can be used to research and analyse potential betting systems and strategies, and to build predictive models or ratings to assess comparative chances of competitors in a sporting event.

We have developed an evaluation tool which uses metrics inspired by the financial sector for quantifying strategy performance. Moreover, an optimisation platform enables strategy parameters to be automatically refined using a stochastic search heuristic in order to improve strategy performance. The method adopted is capable of optimising problems that have a varying number of both discrete and continuous parameters. It is also suited to optimisation problems where little information is known about the optimisation landscape; that is to say where neither the optimal solution is known, nor a systematic method for finding it.

Finally, in order to demonstrate the power of our implementation, we conduct analysis on a number of simple and complex betting exchange trading strategies.

1.4 Outline

The remainder of this dissertation is organised as follows:

Chapter 2 describes the background theory to the work presented in this thesis. The topics of betting, betting exchanges, markets, and trading strategies are introduced, with examples. Previous research in related areas is surveyed and lastly the notions and the importance of event-driven programming and strategy optimisation are presented.

Chapter 3 introduces the Universal Betting Exchange Language (UBEL) and our complex event processing system. UBEL is an object-oriented functional programming language for defining betting strategies using as basis the UrbiScript programming language. We begin by describing UrbiScript itself and by highlighting the ways in which it can be enhanced. We show how we have adapted it to the domain of betting rather than its traditional application domain of robotics. Finally we present a complex event processing system which is based on UrbiEngine.

Chapter 4 presents the SPORTSBET architecture and an overview of the framework. This chapter focuses on how the major elements and components within the framework are used by, or interact with, other major elements and components within the framework. We discuss the series of decisions we took based on a wide range of factors which resulted in this architecture and how each of these decisions impacts on the quality, performance and maintainability of the system.

Chapter 5 presents the optimisation platform. A detailed description and implementation of the optimisation algorithm adopted is shown. The user can apply the optimisation algorithm to optimise the parameters of a given strategy over a set of markets. In addition the user can select the fitness function for the strategy evaluation. The termination criteria of the optimisation algorithm are tunable, thus providing the user with potential to tailor the optimisation towards speed or accuracy. Finally we show how Walk-Forward Analysis is adopted in order to avoid overfitting.

Chapter 6 discusses the implementation of SPORTSBET. We research deeper into the specifics of implementing a framework for automatic betting. We discuss what betting exchanges APIs (Application Programming Interfaces) services are, the different services available, and ways of calling those services using UBEL. We describe how SPORTSBET supports those services and adds more functionality to them. Finally, we present some of the basic features of SPORTSBET.

Chapter 7 presents numerical results produced using SPORTSBET. We explain and evaluate different betting exchange trading strategies of increasing levels of complexity.

Chapter 8 concludes the thesis by summarising and evaluating the achievements presented and highlighting opportunities for future work.

1.5 Statement of Originality and Publications

I declare that this thesis was composed by me alone, and that the work that it presents is my own except where otherwise stated.

The following publications arose from work conducted during the course of this PhD.

- **SPORTSBET: A Tool for the Quantitative Evaluation and Execution of Betting Exchange Trading Strategies**. *Proc. 8th International Conference on Quantitative Evaluation of Systems (QEST 2011)*, Aachen, Germany, pp. 155-156, September 2011. Material from this paper appears in Chapters 1, 4, and 7.

- **An Environment for the Specification and Evaluation of Betting Strategies**. *Proc. 3rd IMA Intl. Conference on Mathematics in Sport (IMA 2011)*, Salford, UK, pp. 251–257, June 2011. Material from this paper appears in Chapters 1, 3, 4, 6 and 7.
- **Metaheuristic Optimisation of Parameterised Betting Exchange Strategies**. *Proc. 4th IMA Intl. Conference on Mathematics in Sport (IMA 2013)*, Leuven, Belgium, pp. 369–375, June 2013. Material from this paper appears in Chapters 3, 5 and 7.

Chapter 2

Background Theory

This chapter presents the background theory underlying the work described in this thesis. A general overview of betting models is provided, before considering the betting exchange model in more detail. This is followed by a discussion of betting strategies from which the main characteristics of a strategy can been seen. Subsequently, previous research in related areas is surveyed and the chapter concludes by describing the theories of event-driven programming and stochastic optimisation, and some of their applications.

2.1 Introduction to Betting Markets

2.1.1 History of Sports Betting

Sports betting has been around for as long as the history of sports themselves [86]. Whenever the first bet was made, it started off a chain of events making gambling and betting a central part of many cultures and, since the beginning of the 21st century, it has evolved into a multi-billion pounds industry involving both legal and illegal sports betting [62] (see Figure 2.1).

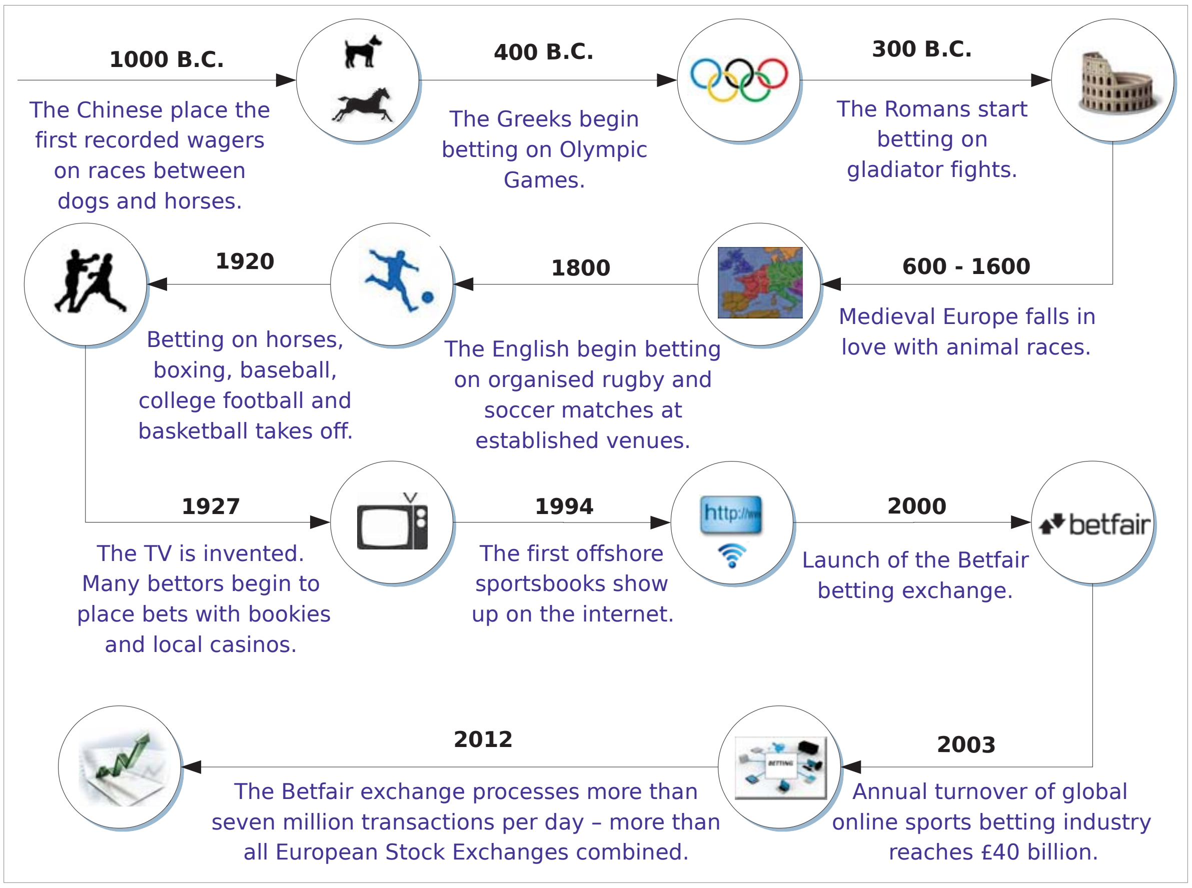

1000 B.C.
The Chinese place the first recorded wagers on races between dogs and horses.
400 B.C.
The Greeks begin betting on Olympic Games.
300 B.C.
The Romans start betting on gladiator fights.
600 - 1600
Medieval Europe falls in love with animal races.
1800
The English begin betting on organised rugby and soccer matches at established venues.
1920
Betting on horses, boxing, baseball, college football and basketball takes off.
1927
The TV is invented. Many bettors begin to place bets with bookies and local casinos.
1994
The first offshore sportsbooks show up on the internet.
2000
Launch of the Betfair betting exchange.
2003
Annual turnover of global online sports betting industry reaches £40 billion.
2012
The Betfair exchange processes more than seven million transactions per day – more than all European Stock Exchanges combined.

Figure 2.1: History of betting (adapted from `http://visual.ly/history-betting`)

The advent of sports books and online sports betting has resulted in a wealth of money taken in by the sports betting industry. Sports betting is simply the act of placing an amount of money (a wager), as a bet on a predicted outcome of any sporting event. Wagers can be placed on any number of possible outcomes of a sporting event. This can include wagering on a team to win a volleyball match, or a player to win a set in a tennis match, or even the final score in a football match. Moreover when a sport involves multiple outcomes, such as horse racing, one can place bets on first, second, and third place.

In the beginning betting was an activity that took place between two individuals and soon evolved into an activity occuring between an individual and a betting organisation (Bookmaker model). For many years that was the accepted norm. Then, in the late 1990s, Andrew Black came up with the idea of using the stock exchange model to operate betting markets. This model (Betting Exchange) allows customers to buy and sell, or back and lay, sporting events outcomes more or less in the same manner as people buy and sell shares. Although bets are still mediated through an organisation, such as Betfair, the odds are set by other individuals, essentially person-to-person betting. Before Betfair, there were two main ways to bet on sports. Firstly via **Pool-betting operators**, where a number of people bet on an event, creating a pool of money. The operator of the pool takes a cut (usually upwards of 20 per cent) and then gives out the rest of the money to the customers who chose the winning selection. And secondly via **Fixed-odds Bookmakers**, where the bookmakers offer their own odds on any range of sporting events [64]. The essentials of these models are recapped in the following subsections.

2.1.2 Bookmaker model

Traditionally, betting markets have been run by a closed community of licensed dealers, known as bookmakers (or bookies) [31]. Bookmakers take bets on sporting and other events at agreed upon odds, and perform a function similar to that of market makers in financial markets in the sense that facilitate liquidity by continuously quoting prices at which they are prepared to deal [103, 14, 65]. In betting the most common type of bet is known as a fixed-odds bet. Suppose player A wishes to back (bet on) some outcome and player B wishes to lay (bet against) the same. Then under a fixed-odds bet, A pays B a certain amount, which is known as his stake, before the event happens; then if the event materialises, B pays A his stake back plus a payout of the stake multiplied by the pre-agreed odds.

For example, A places a bet of £100 at the agreed odds of 3:1 or "Three to One" that Arsenal will win the Champions League. In case Arsenal succeeds, then A collects £300 from B as well as a refund of his original stake; otherwise B keeps the £100 stake.

Odds for different outcomes in a single bet are presented either in European format (decimal odds), UK format (fractional odds), or American format (moneyline odds) [42].

- **Fractional Odds**: The ratio of the amount won to the stake. Odds of 4:1 or "Four to One" would imply that for every £1 staked you stand to win £4. When the first figure is higher than the second it is called "odds against". When the first figure is smaller than the second it is called “odds on” and the amount of winnings will be smaller than the stake. For example if the odds offered were 1:4 then for every £4 staked you would stand to win £1.

- **Decimal Odds**: The ratio of the full payout to the stake, in a decimal format. The decimal odds of an outcome are equivalent to the decimal value of the fractional odds plus one. So decimal odds of 5.0 are equivalent to fractional odds of 4:1.
- **Moneyline Odds**: There are two possibilities: the figure quoted can be either positive or negative. When positive the figure reflects the amount won on a stake of 100 currency units. When negative the figure reflects the stake needed to win 100 currency units. For example, a moneyline of +400 is equivalent to fractional odds of 4:1, while a moneyline of -400 is equivalent to fractional odds of 1:4.

When betting with bookmakers, customers are restricted to backing outcomes only; the bookmaker plays the role of player B, taking the lay side to every bet. In any sporting event, every outcome has a chance or likelihood of taking place; the odds are simply an interpretation of those chances. For instance, odds of 3:1 imply a view that Arsenal is regarded as three times more likely to fail than to succeed (a 25% probability of Arsenal victory). Bookmakers rely on in-house experts to assess the likelihood of different outcomes and to compile a set of odds in order to balance the books. Odds are described as fair when the implied probabilities of all the possible outcomes in a market sum to one (or 100%). This means that if player A backs all the possible outcomes, then (s)he will get as a return the complete amount of the stake (s)he used for betting. So, in order to make profit, a bookmaker prices up a book so that the implied probabilities of all the outcomes is greater than 1 (or >100%). The excess percentage is known as the “Overround” (in the UK) or “Vigorish” (in the US) and it is the reason for the long-term profit of the bookmaker. Generally bookmakers are interested in securing a stable income through the margin, a percentage of the total stakes placed with them. The expected margin (gain) of a bookmaker on an event with n outcomes can be represented as [74]:

$$E(M) = 1 - \sum_{i=1}^{n} P_i \cdot W_i \cdot d_i \tag{2.1}$$

According to Equation 2.1, the expected margin (M) on each match depends on the probability associated with each outcome (P_i), the percentage of bets on each outcome (W_i) and the quoted odds (d_i), respectively. This implies that bookmakers possess different ways to set their prices [74]. They can try to forecast accurately game outcomes so that the odds reflect these expectations, or they can try to forecast the distribution of bets on each outcome. Alternatively, it is possible to use some combination of the two approaches discussed previously. Equation 2.1 implies that in order to calculate the actual margin that a bookmaker earns from a single match, we need to know both the odds on every outcome and the distribution of bets across outcomes. Although information on quoted odds is publicly available, this is not the case for the distribution of bets. This means that we cannot calculate the actual margin of bookmakers but rather an implied margin, denoted as $M^{'}$. According to the literature [119], this is estimated by assuming that the bets are equally distributed across outcomes and that the odds are set according to the true probabilities. In practice, bookmakers employ odds compilers who have special knowledge of specific sports in order to estimate the true probability of each possible outcome. Assuming that odds are set on the basis of these true probabilities, then the fair odds on an outcome i is simply the reciprocal of the probability P_i of the occurrence of that outcome. However, if odds were priced exactly at their fair level according to the true probabilities then the expected bookmaker gain would be zero. For this reason, actual implied odds are somewhat smaller than fair odds in order to allow a positive margin for bookmakers. Accordingly, actual implied odds do not correspond to true probabilities but to somewhat larger implied probabilities (denoted $P_i^{'}$). The expected implied margin can be estimated as [119]:

$$E(M^{'}) = (\sum_{i=1}^{n} P_i^{'}) - 1 = (\sum_{i=1}^{n} \frac{1}{d_i}) - 1 \tag{2.2}$$

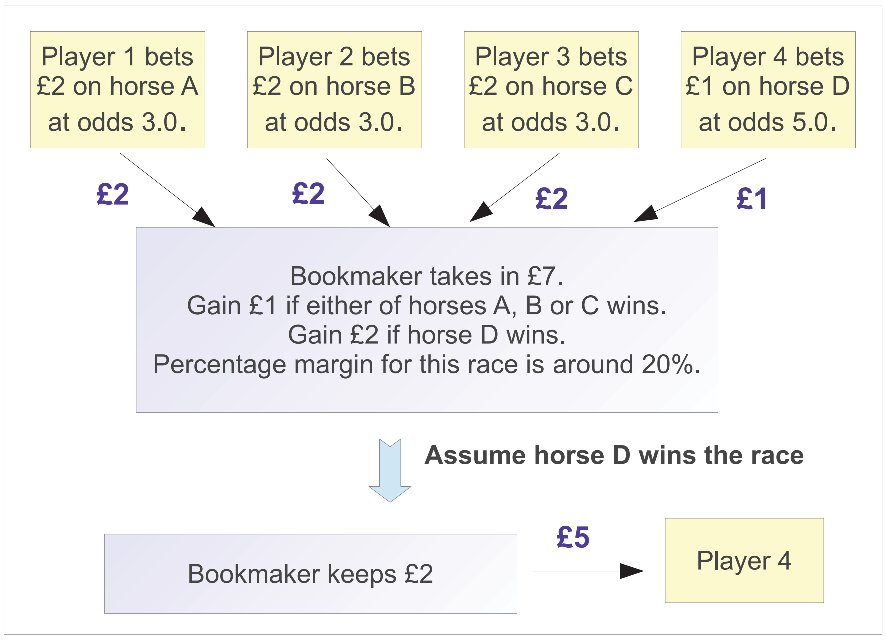

Player 1 bets £2 on horse A at odds 3.0.
Player 2 bets £2 on horse B at odds 3.0.
Player 3 bets £2 on horse C at odds 3.0.
Player 4 bets £1 on horse D at odds 5.0.
£2
£2
£2
£1
Bookmaker takes in £7.
Gain £1 if either of horses A, B or C wins.
Gain £2 if horse D wins.
Percentage margin for this race is around 20%.
Assume horse D wins the race
Bookmaker keeps £2
£5
Player 4

Figure 2.2: Bookmaker model

Throughout the 1990s, this bookmaking model (see Figure 2.2) was the only business model for betting. In fact, in the UK it was illegal for anyone other than a licensed bookmaker to accept bets and major players like William Hill, Ladbrokes and Coral dominated the market. However, in 2000 the situation changed dramatically with the arrival of betting exchanges.

2.1.3 Betting Exchange model

Betting exchanges allow people with different opinions on the likely outcome of an event to bet against each other. Betfair is the most well known example, but there are also others, such as Betdaq (recently purchased in January 2013 by Ladbrokes PLC), WBX (World Bet eXchange), Smarkets and Matchbook. Betting exchange markets are basically dynamic markets of fixed-odds bets made directly between bettors, thus eliminating the need for a bookmaker (see Figure 2.3). This means that exchange participants can not only back event outcomes, as supported in the standard bookmaking model, but can also lay (i.e. bet against) event outcomes. For the reason that individuals are generally less cautious in terms of the odds they offer than professional bookmakers, and due to the fact that the opinions of a group of individuals is likely to vary more than those of a group of bookmakers, the prices on offer on a betting exchange are almost always more favourable than those available from bookmakers. On the other hand, in a betting exchange market market liquidity may be limited and so bets placed may not be matched, whereas bookmakers may be willing to accept large bets in full. Betting exchanges also do not (yet) allow customers to create custom bets whereas bookmakers may be willing to take such bets.

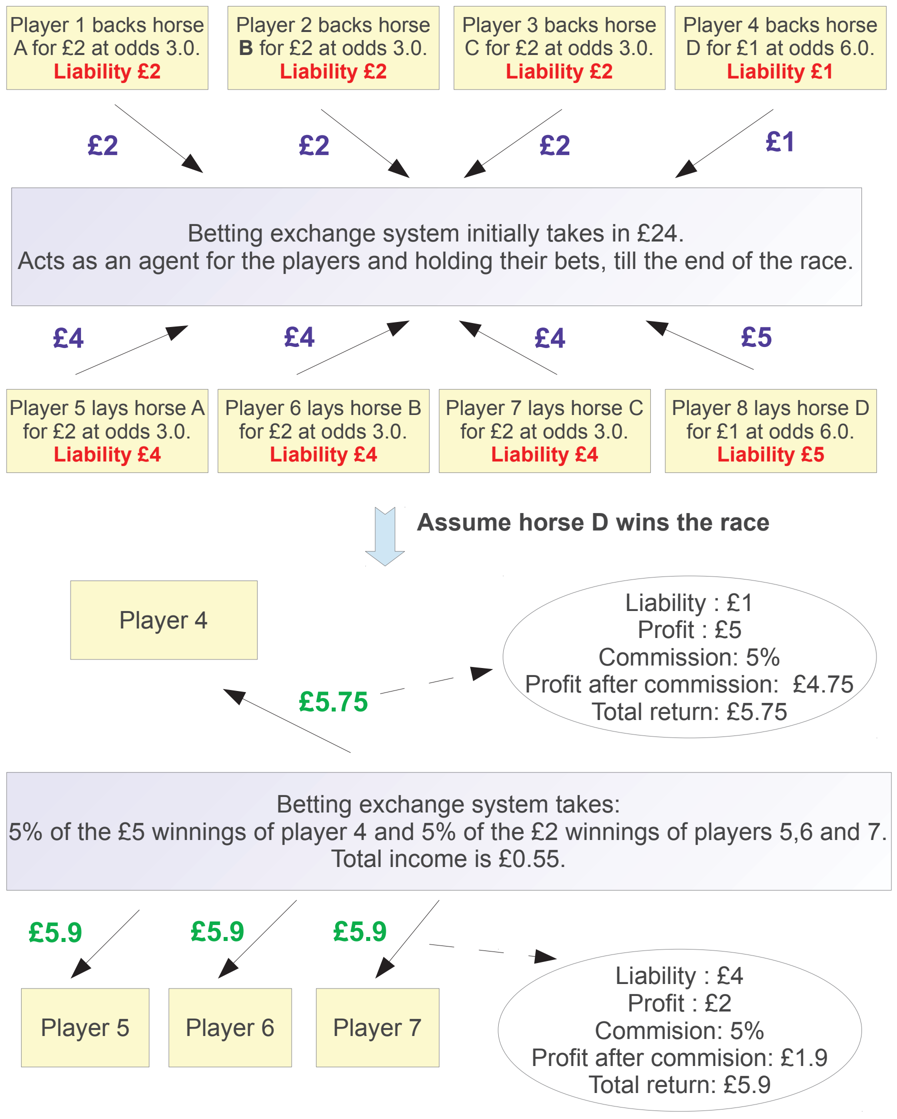

Player 1 backs horse A for £2 at odds 3.0. Liability £2
Player 2 backs horse B for £2 at odds 3.0. Liability £2
Player 3 backs horse C for £2 at odds 3.0. Liability £2
Player 4 backs horse D for £1 at odds 6.0. Liability £1
£2
£2
£2
£1
Betting exchange system initially takes in £24.
Acts as an agent for the players and holding their bets, till the end of the race.
£4
£4
£4
£5
Player 5 lays horse A for £2 at odds 3.0. Liability £4
Player 6 lays horse B for £2 at odds 3.0. Liability £4
Player 7 lays horse C for £2 at odds 3.0. Liability £4
Player 8 lays horse D for £1 at odds 6.0. Liability £5
Assume horse D wins the race
Player 4
£5.75
Liability : £1
Profit : £5
Commission: 5%
Profit after commission: £4.75
Total return: £5.75
Betting exchange system takes:
5% of the £5 winnings of player 4 and 5% of the £2 winnings of players 5,6 and 7.
Total income is £0.55.
£5.9
£5.9
£5.9
Player 5
Player 6
Player 7
Liability : £4
Profit : £2
Commision: 5%
Profit after commision: £1.9
Total return: £5.9

Figure 2.3: Betting exchange model

Exchanges allow customers to place bets “in-play”. As the name suggests, in-play betting is betting while an event is in progress, right up until the end of the contest. Betting exchanges have pioneered in-play betting and offer a range of in-play opportunities in various sports such as soccer, basketball, tennis, horse racing, and more. In the case of horse racing, betting generally remains open until the first horse crosses the finishing line. In the case of a photo finish (i.e. multiple competitors cross the finishing line at nearly the same time), betting on which horse has won remains open until the stewards make their decision. Proponents argue that this has created a significantly more exciting betting experience. Detractors argue that in-play betting has facilitated new opportunities for organised crime and corrupt sports persons to anonymously perpetrate betting scams[1].

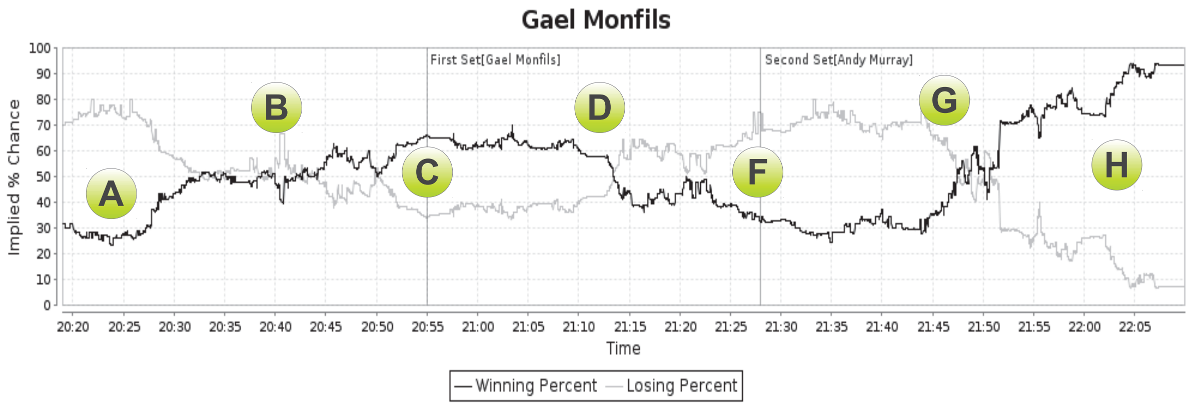


Figure 2.4: Visualisation of in-play odds evolution in the Match Odds market over the course of a single match (Monfils v. Murray)

Figure 2.4 shows an example of the odds evolution in a tennis match between Gael Monfils and Andy Murray in BNP Paribas Masters 2010 tournament which is part of the ATP World Tour Masters series. The graph shows how the price of Gael Monfils evolves throughout the game. Gael Monfils won the match 2 sets to 1 (6-2, 2-6, 6-3). In the beginning (A) of the match, the back price is stable of around 3.25 (30.77% winning percentage), which means Andy Murray is the favourite. As the match goes on (B), the price fluctuates as the momementum swings back and forth. Next, Monfils gets the upper hand and wins the first set 6-2 (C). The price falls and stabilises on a level about 1.6 (62.5% winning percentage), which means Gael Monfils is now the favourite. However, Murray fights back (D), winning the second set 2-6 (F) and becomes again the favourite. The third set is a tight affair, with Monfils breaking his opponent to go up 3-2 (G) and then goes on and win the third set 6-3 and thereby the match (H).

Compared to bookmakers' odds, exchange prices, especially for popular events, are more likely to reflect the true odds better and be more competitive. Typically, the overround of a liquid ex-

[1]For a comprehensive and insightful overview of the controversies sparked by the betting exchange industry, interested readers might like to read Niall O'Connor's "A History of the Betting Exchange Industry." available at `http://www.bettingmarket.com/refraichir010388.htm`

change market will be close to zero, so bettors get more value for money than with a traditional bookmaker, even taking into account the commission on profit paid to the exchange.

Real example: *"Deportivo La Coruna vs Barcelona"*

Table 2.1: Overround in betting models

Selection	**Bookmaker (William Hill)**	**Exchange (Betfair)**
Home	4.40	5.8
Away	1.60	1.66
Draw	3.60	4.3
Total (%)	113.005	100.7
Overround (%)	13.005	0.7

So according to Table 2.1 the overround for the bookmaker William Hill is 13.005%. That means if the bettor backs all the selections offered, he will lose 13.005% of his stake. On the other hand the overround for the corresponding betting exchange markets in Betfair is 0.7%. Comparing the two overrounds it is clear that placing bets in a betting exchange environment offers better value[2].

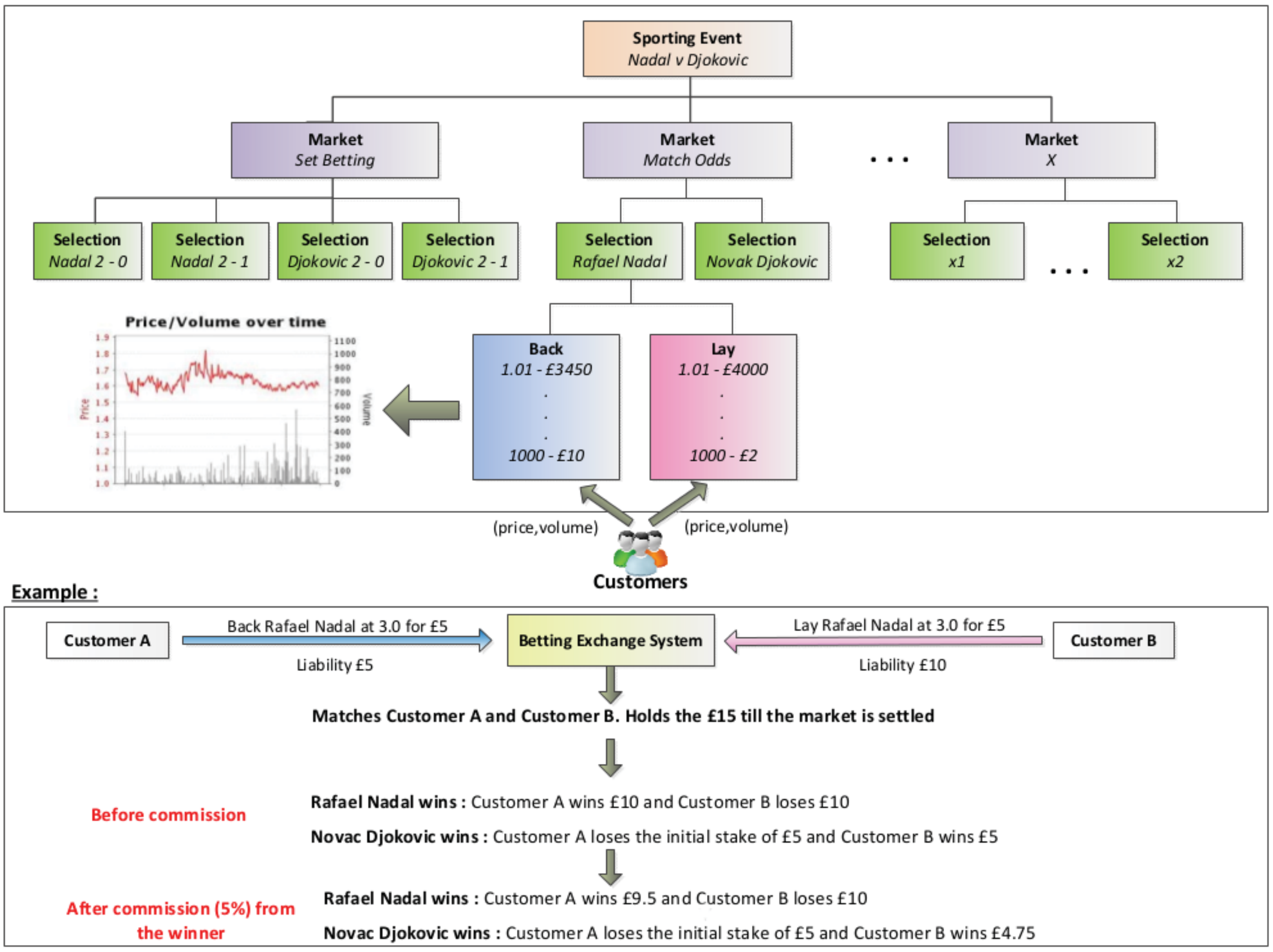
The real difficulty for exchanges has been to achieve sufficient liquidity. Market liquidity refers to how much money is available to be matched in a market and is going into a market at a specific time. Some markets are of high liquidity while others are of low liquidity, depending on the popularity of the event in question (which in turn depends on factors such as the extent of media coverage), as well as the popularity of the particular market and the corresponding betting exchange platform. Betfair was one of the first betting exchanges, and is the world's largest financial exchange in terms of trade frequency and global reach. This means liquidity levels are generally high in Betfair markets. Certain competitors such as Betdaq are also starting to attract significant liquidity in selected markets (particularly those related to horse racing).

[2]This comparison ignores betting exchange commission, typically 5% of net profits in a given betting exchange market; but it is virtually always the case that the same conclusion holds even when this is taken into account.

Betting exchanges have a very efficient system which aggregates the stakes of backers and layers into pools which can be matched. As an example if 3 separate players offered odds of 5.0 against a horse with stakes of £100, £200 and £300 this would be shown as £600 available to back at 5.0, of which a player can back any amount up to £600 [72].

In general, there are multiple betting markets associated with a particular sporting event. As an example a football match can have a *Match Odds*, a *Correct Score*, a *Half Time Score* and as well as many other markets. Markets are composed of selections, which customers may back or lay. The selections reflect the possible outcomes of the market.

Figure 2.5 shows the structure of the betting exchange model.



Matches Customer A and Customer B. Holds the £15 till the market is settled

Rafael Nadal wins : Customer A wins £10 and Customer B loses £10

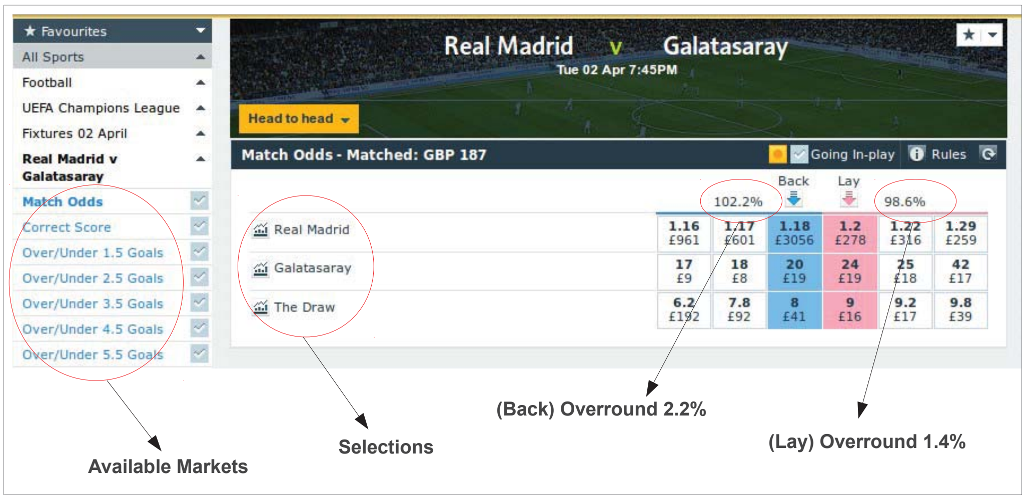
Novac Djokovic wins : Customer A loses the initial stake of £5 and Customer B wins £5

Rafael Nadal wins : Customer A wins £9.5 and Customer B loses £10

Novac Djokovic wins : Customer A loses the initial stake of £5 and Customer B wins £4.75

Figure 2.5: Betting exchange model structure

Figure 2.6 is a screenshot of a typical Betfair market, specifically a *Match Odds* market, for the football match *"Real Madrid vs Galatasaray"*. On the left side of the screenshot you can see all the markets associated with this particular match. For sporting events, Betfair shows the possible outcomes (Selections), and columns entitled “Back” and “Lay”. The “Back” column shows the odds available to a player who wants to back an outcome and how much money is available at those odds. The “Lay” column shows the odds available to a player who wants to lay an outcome and how much money is available at those odds. The blue highlighted boxes shows the best available prices and volumes for backing the selections, while the pink highlighted squares shows the same information for laying the selections. Moreover, if a user does not agree with the current available odds, (s)he can very easily place their own odds and wait for their bet to be matched.



Match Odds - Matched: GBP 187

	102.2%		Back	Lay	98.6%	
Real Madrid	1.16 £961	1.17 £601	1.18 £3056	1.2 £278	1.22 £316	1.29 £259
Galatasaray	17 £9	18 £8	20 £19	24 £19	25 £18	42 £17
The Draw	6.2 £192	7.8 £92	8 £41	9 £16	9.2 £17	9.8 £39

Figure 2.6: Screenshot of a Betfair Match Odds market for the match Real Madrid against Galatasaray

2.2 Financial and Betting Strategies

Exchange markets have many similarities with the financial markets [92, 83]. In financial markets traders talk about operating on “both sides of the market”, meaning that someone is buying and selling. In this way, people can take part in trading where low-risk profits are guaranteed by buying low and selling high. Being able to back as well as lay on betting exchanges allows you to do the same thing in betting markets.

2.2.1 Arbitrage and Trading

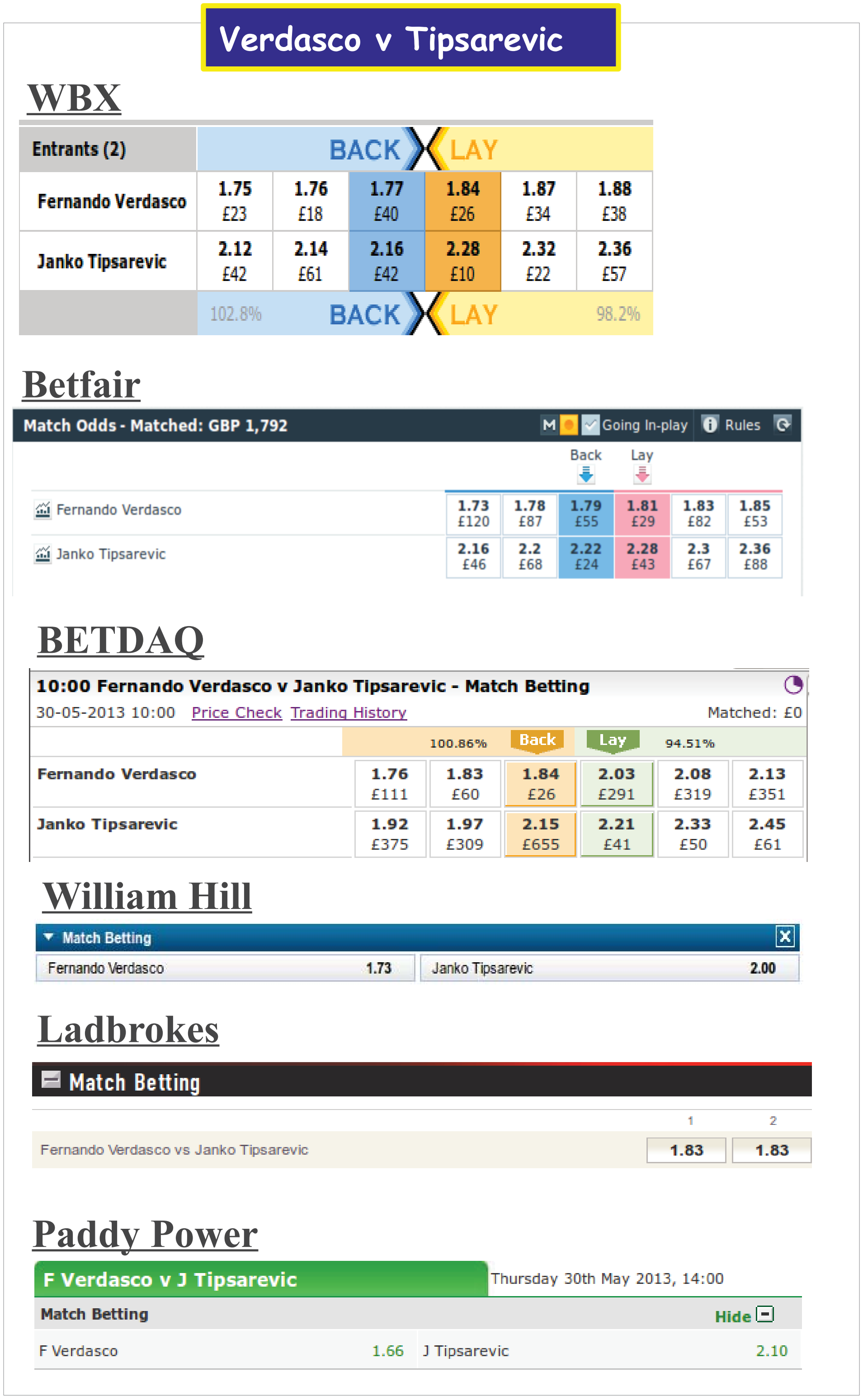
Arbitrage [47, 79] is a very old and well known method of profiting from pricing anomalies. Specifically, it involves simultaneously buying a security at a lower price in one market and selling the same security at a higher price in another market to make a profit on the spread between the prices. As an example, if General Motors common stock trades at £50 on the London Stock Exchange and at a price equivalent to £49.50 on the New York Stock Exchange, an investor could guarantee a profit by purchasing the stock on the New York Stock Exchange and simultaneously selling the same amount of stock on the London Stock Exchange[3]. Although the price difference may be very small, arbitrageurs typically trade regularly and in huge volume, so that they can make sizable profits.

Similar to financial markets, betting exchanges have opened up a new range of arbitrage possibilities since it is possible to lay as well as to back an outcome [45]. So, arbitrageurs in exchanges attempt to simultaneously bet on all possible outcomes to make a guaranteed profit. Such opportunities rarely occur within a given market on a given betting exchange, but may arise more frequently in equivalent markets hosted by different betting exchanges. The latter is in principle the same as the occasional arbitrage opportunities that arise when comparing back prices across different bookmakers. Arbitrage using back and lay side is possible if a lay bet on one exchange provides shorter odds than a back bet on another exchange or bookmaker. However, the commission charged by the exchanges and bookmakers must be included in calculations. Market payout rules must also be carefully compared to ensure that profits are guaranteed, especially under unusual circumstances such as floodlight failure or player retirement on account of injury. Often apparent arbitrage opportunities turn out to be due to variations in payout rules, with price differences reflecting the risk of the occurrence of some unusual event outcome.

In Figure 2.7 we can see an arbitrage opportunity in the tennis match between Verdasco and Tipsarevic at French Open 2013 without taking into consideration the commission fees applied

[3]Naturally traders must also pay attention to transaction costs and government taxes/duties, which may be different in different markets.

in each betting provider. Table 2.2 shows the pairs of betting providers and where arbitrage opportunities occur. However, in the specific example if the normal commission fee of 5% is applied there are no arbitrage opportunities.



Verdasco v Tipsarevic

WBX

Entrants (2)			BACK	LAY		
Fernando Verdasco	1.75 £23	1.76 £18	1.77 £40	1.84 £26	1.87 £34	1.88 £38
Janko Tipsarevic	2.12 £42	2.14 £61	2.16 £42	2.28 £10	2.32 £22	2.36 £57
	102.8%		BACK	LAY		98.2%

Betfair

Match Odds - Matched: GBP 1,792 M Going In-play Rules

			Back	Lay		
Fernando Verdasco	1.73 £120	1.78 £87	1.79 £55	1.81 £29	1.83 £82	1.85 £53
Janko Tipsarevic	2.16 £46	2.2 £68	2.22 £24	2.28 £43	2.3 £67	2.36 £88

BETDAQ

10:00 Fernando Verdasco v Janko Tipsarevic - Match Betting
30-05-2013 10:00 Price Check Trading History Matched: £0

		100.86%	Back	Lay	94.51%	
Fernando Verdasco	1.76 £111	1.83 £60	1.84 £26	2.03 £291	2.08 £319	2.13 £351
Janko Tipsarevic	1.92 £375	1.97 £309	2.15 £655	2.21 £41	2.33 £50	2.45 £61

William Hill

Match Betting

Fernando Verdasco	1.73	Janko Tipsarevic	2.00

Ladbrokes

Match Betting

	1	2
Fernando Verdasco vs Janko Tipsarevic	1.83	1.83

Paddy Power

F Verdasco v J Tipsarevic Thursday 30th May 2013, 14:00

Match Betting Hide

F Verdasco	1.66	J Tipsarevic	2.10

Figure 2.7: Arbitrage opportunity in the tennis match between Verdasco and Tipsarevic at French Open 2013 – no commission included

Table 2.2: Arbitrage opportunities between different betting providers in a single tennis match – no commission included

	WBX	Betfair	Betdaq	William Hill	Ladbrokes	Paddy Power
WBX	✗	✗	✗	✗	✗	✗
Betfair	✗	✗	✓	✗	✓	✗
Betdaq	✗	✓	✗	✗	✗	✗
William Hill	✗	✗	✗	✗	✗	✗
Ladbrokes	✗	✓	✗	✗	✗	✗
Paddy Power	✗	✗	✗	✗	✗	✗

Trading. Harris [59] defines trading as "Trading is a search problem. Buyers must find sellers and sellers must find buyers. Every trader wants to trade at a good price. Sellers seek buyers willing to pay high prices. Buyers seek sellers willing to sell at low prices. Traders must also find traders who are willing to trade their quantities or sizes they desire. Traders who want to trade large quantities may have to find many willing traders to complete their trades".

Basis Trading [124] in finance is a trading strategy usually consisting of the purchase of a particular security and the sale of a similar security. Basis trading takes place when the trader feels that the two securities are mispriced with respect to each other, and that the mispricing will correct itself so that the gain on one side of the trade will more than cancel out the loss on the other side of the trade.

On a betting exchange, a trader operates similarly to an arbitrageur, but he is willing to take on extra risk and bet on sporting events where no immediate profit is possible. A trader hopes to make a profit by closing out the bet at a later stage at more favourable odds. Trading can be done either before the start of a sporting event or while the sporting event is in progress (if in-play betting is offered), although the latter can be much more risky [123].

Another famous stock strategy is the **Contrarian Stock Selection Strategy** [29, 36]. It consists of buying stocks that are out of favour and selling short stocks that are popular. The strategy is formulated on the premise that the stock market overreacts to news, so winners lend to be overvalued and losers undervalued. So an investor who exploits this inefficiency gains when stock prices revert to fundamental values. This strategy can be very easily adopted in betting exchange markets, under the condition that it consists of buying and selling bets, and that the news would include information such as a *goal being scored* or *match point* (in in-play betting), or a significant player of a team will not play (in pre-play betting).

2.2.2 Trading Systems

A trading strategy is a predefined set of rules for making trading decisions. A trader can be a discretionary trader or a system trader. A system trader uses automatic trading systems in order to execute the trades while in discretionary trading the trader decides which trades to make. Discretionary trading is used most by traders that want to be in control of every trading decision while system trading is used most by traders who want qualities like speed, precision, and accuracy in their trading. Note that it is possible for a discretionary trader to use system trading, while it is impossible for a system trader to use discretionary trading. Many trading systems have been developed covering both types of trading, with the quantitative trading systems being the most advanced and complex ones. A typical structure of a quantitative trading system is shown in Figure 2.8.

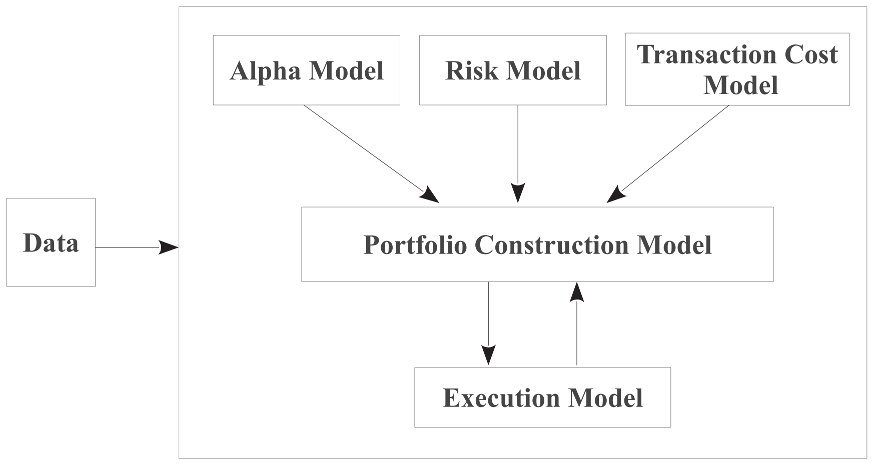


Figure 2.8: Typical structure of a quantitative trading system [88]

It has five modules [88] :

- **Alpha Model**: Is designed to predict the direction of whatever markets the quant has decided to include in his strategy. Alpha Model is focused on making money by actively predicting the future.

- **Risk Model**: Is designed to control the size of desirable exposures or to deal with undesirable types of exposures. In general Risk Models reduce the amount of money a quant can make, but at the same time they can increase the safety of the investment.

- **Transaction Cost Model**: Is designed to determine the cost of the trades needed to jump from the current portfolio to a new one desired by the portfolio construction model.

- **Portfolio Construction Model**: Is designed to determine what portfolio the quant wants to own. As an input it takes the arguments of the "optimist" (alpha model), those of the "pessimist" (risk model) and those of the "cost-conscious accountant" (transaction cost model). The output is what proportion of capital should be ideally invested in each candidate investment.

- **Execution Model**: Is designed to implement the portfolio decisions made by the portfolio construction model. There are two ways of trade execution: electronic or through a human intermediary.

Quant traders generally build their models that take input data, make some calculations utilising this data and then process the resulting trading decisions. Moreover, using these data the quant can perform research, which usually involves some form of testing and simulation. Note that the structure shown in Figure 2.8 is not universal as many quant strategies may run without some of the modules (e.g. transaction cost model, portfolio construction model) [88].

Concerning the similarities (mentioned in previous sections) between financial stock markets and betting exchange markets, it becomes obvious that the whole concept of trading strategies, used in financial trading, can be adopted in the area of betting exchanges.

2.2.3 Trading Strategies

As Ford [44] points out “Strategies are all about probability. The aim when creating a strategy is to have a higher probability of profit than loss. It is impossible to eliminate the losses of a strategy, so the aim is to minimise them”. A trading strategy is a set of rule-based systematic trading instructions which determines the optimal holding of each security at a particular time. Strategies are a fundamental part of a successful trading system. A sports betting strategy can accept as input market prices and volumes, underlying sporting event data, analyst opinion, model output and existing positions. In addition there are other important considerations when trading: How much to trade? How should capital be allocated between trades? How to manage risks (e.g. stop losses and take profits)? How to execute trades?

Sports betting markets share similar characteristics with stock markets. The finance literature hypothesize that stock markets are efficient, which means that the market prices fully reflect all the public available relevant information. Since the first paper on the efficient markets hypothesis [40], there has been a lot of debate in the literature regarding the efficiency of information markets, including both financial and sports betting markets. Sport betting markets can swing between market efficiency and inefficiency phases [82]. A sport betting market is said to be efficient if market prices would at all times fairly reflect all known information for the corresponding sporting event. That implies that it is impossible to make economic profits by trading on the basis of this public information. However, faster access to this public information can make a trading strategy more profitable (Latency based advantage). Moreover, literature has shown that there are inefficiencies in sports models. Maher [81] assumed independent Poisson distributions for home and away goals, Dixon and Coles [35] took this idea further by accounting for fluctuations in performance of individual teams and estimation between leagues. Dixons and Coles model showed that certain scores (e.g. low-scoring draws) were systematically mispriced by the Maher model. The Dixon and Coles model is a simple and robust full-time score model, but not all of its assumptions are met. These mathematical models they do not take into account external information about match circumstances such as: injuries, motivation, newly signed players. That means mathematical models cannot model team news (unless this is incorporated into the model somehow). Because of that profitable opportunities emerge within sports betting markets that feature participants who use the "inefficient" sports model. There are many academic examples of those inefficiencies (Terrell and Farmer, 1996, greyhound racing) [109], (Williams and Paton, 1998, horse racing) [126] and (Busche and Walls, 2000, horse racing) [17].

The flexibility to back and lay a specific sporting event opens up many betting and trading opportunities. By combining many different bets, perhaps on different markets, it is possible to form sophisticated betting strategies that have the potential to show a profit over many sporting events.

- ***Trading***

  Suppose there is a tennis match taking place between Djokovic and Stepanek. Djokovic is available to back at 1.09 and lay at 1.1. Now assume that a bettor has decided to lay Djokovic at 1.1 for £100 (win £100 if Djokovic loses, lose £10 if Djokovic wins). Suppose that Djokovic lose the first set. The market reacted on that and Djokovic can now be backed at 1.8. The bettor now decides to back him for £60 at 1.8 to lock in a profit. If Djokovic wins, the bettor will win £60 at odds of 1.8 = £48, minus the losing initial lay of £100 at 1.1 = £10. So the total profit will be £38. If Stepanek goes on to win, the bettor will win the initial bet of £100 (lay Djokovic), minus the losing bet of £60. So the total profit will be £40. Therefore, as we can see, that a profit of either £38 or £40 is guaranteed (before Betfair's commission of between 2% and 5% of any net profits you make in a given market; if you lose you do not pay any fees).

  The above bet example is a simplified but realistic example of the strategy of a trader. A strict implementation of the above strategy can be defined like this:

  In all tennis matches between Djokovic and Stepanek:

  1. If in the market *Match Odds* the odds to lay Djokovic are ≤ 1.1 and the odds to back him are ≥ 1.09 lay him for £100

2. If the bet in 1 has been placed and matched, and in the market *Match Odds* the odds to back Djokovic increase by 71% or more then back him for £60.

- ***Strategy "10 minutes no Goal"***

  In betting exchanges there are many markets associated with a football match. One of them is to bet on in which minute the first goal will be scored (*First Goal*). So a strategy can be:

  In all football matches of Premiership for the season 2008/09:

  – If 30 minutes before kick-off, in the market *Match Odds* the difference between the back odds of the home team win and the away team win is < 1.5, then in the market *First Goal*, lay the selection "0 - 10 minutes" for £10.

  So what exactly this strategy suggests is that in a football match if there is no heavy favourite, then it would likely be a tense match with both sides looking to keep it tight in the first 10 minutes. So both teams will take few chances early on and is more possible that there will be no goal in the first 10 minutes.

- ***Strategy "Lay the Draw"***

  In all football matches that in-play betting is allowed:

  1. If the lay odds for the draw in the market *Match Odds* are between 3.0 and 5.0, then lay the selection "draw" for £50.
  2. If the bet specified in 1 has been placed and matched, then if in the market *Correct Score* the selection "0-0" can be backed at odds $\geq$10.00 then back this selection for £13.50.
  3. If a goal is being scored during the game then in the market *Match Odds* if the odds to back the selection draw are > 5.0 then back that selection for £25.

- ***Strategy "Stop Loss"***

  Stop losses strategies are used by traders to protect their bankroll and make sure that their losses will not run out of control. The idea is that at the time you place your bet, you give an order to close out the bet if the market price simply runs away from the initial bet and the spread reaches a certain level. This means we should never lose more than a set amount on a trade.

- ***Strategy "Let wins run"***

  "Let wins run" strategies are used by traders in order to increase their long term profits. The idea comes from stock trading where a frequent mistake of traders is to sell a successful stock too soon [23]. To prevent this while simulataneously guarding against loss of accumulated profit, trailing stops are used. For example, let us say we have backed Nadal at 1.9 and the price is now 1.7. We can say if the price reaches 1.8 (trailing stop point) we will close the bet and take the profit we have at 1.8. However, if the price continues to drop, let us say to 1.6 we can move our trailing stop point to 1.7, and so on.

2.2.4 The Trading Strategy Development Process

The development of a trading strategy is a complex process consisting of a number of different stages. The basic steps for the development and application of a trading strategy are:

1. Formulation.
2. Specification in computer-testable form.
3. Testing.
4. Optimisation.
5. Evaluation of performance and robustness.
6. Real time trading.
7. Monitoring of trading performance.
8. Refinement and evolution.

So as an example a trader who wants to test the strategy "*Lay the Draw*", first (s)he must translate the trading rules of the strategy into a computer-testable language. Then he must create a historical simulation of the trading strategy and evaluate it. If the strategy seems promising then optimisation techniques can help to uncover the full scope of its profitability. If the trading strategy looks sound, it can be traded in real time. Furthermore, the trader is able to continually refine the trading rules and repeat the process from the beginning [93].

2.3 Previous Research

As mentioned before financial and betting exchange markets have a lot of similarities. Ramazan [53, 94] states that "Financial traders test historical data to establish specific rules for buying and selling securities with the objective of maximising profit and minimising risk of loss. Traders base their analysis on the premise that the patterns in market prices are assumed to recur in the future, and thus, these patterns can be used for predictive purposes. The motivation behind the technical analysis is to be able to identify changes in trends at an early stage and to maintain an investment strategy until the weight of the evidence indicates that the trend has reversed. Traders and researchers in stock marketing often hold some private trading strategies. Evaluation and optimisation of their strategies is a great benefit for them before they take any risk in realistic trading". This means that stock traders can take less risk if they back-test their trading strategies in house first. Financial researchers often iteratively evaluate their trading algorithms before publishing them. However, it is very hard for traders and researchers to build a system by themselves in order to evaluate their strategies. Another difficulty for them is that it is hard to get huge amount of real stock data for testing.

Many tools have been developed in order to address those needs. The main objectives in building these tools are to provide financial traders, researchers or even data miners with a flexibly and automatically practical infrastructure. With this infrastructure, they can plug in their algorithms easily, and focus on improving the performance of their algorithms using simulation and evaluation techniques on a large amount of historical or real stock data from international markets. All other work, including user interface generation, data preparation, and resulting output must maintained by the tools. Such a tool is F-TRADE (Financial Trading Rules Automated Development and Evaluation).

2.3.1 F-TRADE case study

F-TRADE [20, 19, 18] is an agent service-based automated enterprise infrastructure. It supports back-testing, simulation, evaluation and optimisation of trading strategies and data mining algorithms with online connection to huge amounts of stock data. General F-TRADE has been an online test platform for research and application of multi-agent technology, and data mining in stock markets.

In order to support all these services, F-TRADE allows plugging in of data sources, data requests, trading or mining algorithms and system functional components (known as plug-and-play). From a high-level perspecftive, F-TRADE looks like an online services provider. Figure 2.9 shows its architecture [20, 19, 18].

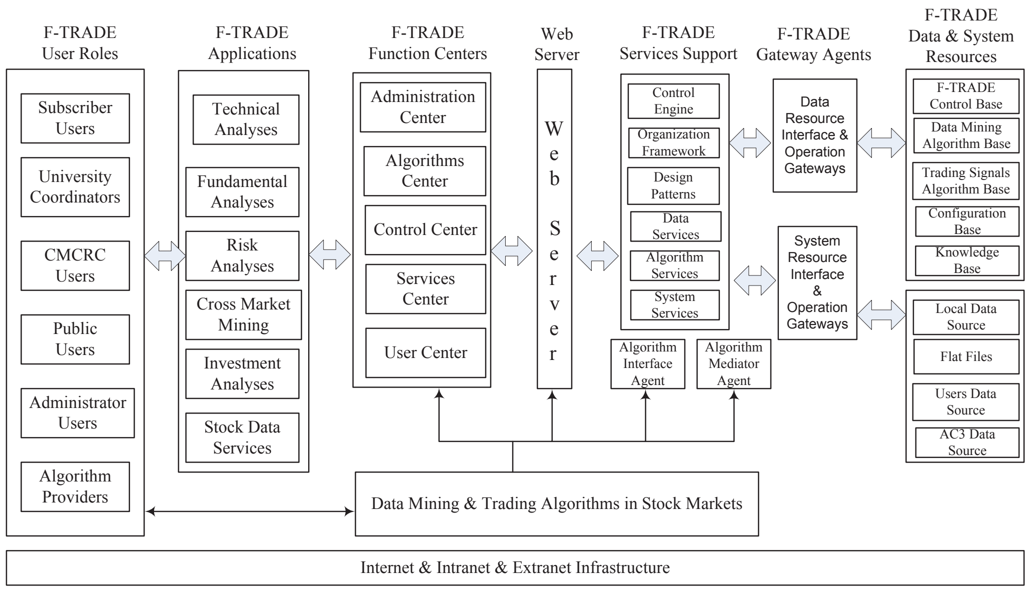


Figure 2.9: The Architecture of F-TRADE [20]

2.3.2 Financial Programming Languages

Functional Programming and Formal Semantics in Finance

Jones et al. [68] at 2000 introduced a combinatory library that allows the description of financial contracts precisely, and a compositional denotation semantics that says what such contracts are worth. Their work has a great impact in financial and computer science areas, as they proved that a lot of the insights useful in the design, semantics and implementation in programming languages are applicable directly to the description and evaluation of financial contracts. The implementation of the combinatory library has been done in Haskell; however, the design is absolutely not Haskell-specific.

The Risla Language

Arnold [5] states "Among the most exciting banking activities is the inter-bank trade of interest rate products. Large amounts of money are transferred, in order to fulfil the bank's current and future financial needs. However, interest rates can change at any moment, and trade in financial products is not risk-free". In order to deal with the problems of long time-to-market and potentially inaccurate implementations, the programming language RISLA (for Rente Informatie Systeem Language —Interest rate information system language) was born. RISLA was developed by the Dutch bank MeesPierson, together with software house Cap Gemini as a specific language for describing interest rate products. RISLA was to be readable for financial engineers, and descriptions in this language were to be compiled into COBOL. It is based on a number of built-in data types for representing cash flows, rates, dates, intervals, etc. In addition it has a large number of built-in operations which allows the manipulation of the data types mentioned above. A product definition specifies the contract parameters, information methods, and registration methods. RISLA is translated into COBOL. Other systems in the bank can invoke the generated COBOL to create new contracts, to ask information about existing contracts, or to update contract information. The initial version of RISLA was used to define about 30 interest rate products [34, 117].

Languages for Betting Exchange Trading Strategies

Although a large volume of literature is available on gambling, the topic of betting exchanges is still under development. This is because most of the research on gambling is concerned with the efficiency and bias of fixed-odds betting markets, or optimal methods of playing certain games with an element of chance. Many tools have been developed for forecasting such as EDDIE (which stands for Evolutionary Dynamic Data Investment Evaluator) [111, 112, 113] but there is nothing available, to our knowledge, for definition, simulation, live execution, evaluation and optimisation of betting exchange strategies. We anticipate that, similarly to financial contracts, betting strategies can also be described and evaluated by either using domain-specific programming languages, or by creating a formal description.

2.4 Event-Driven Programming

2.4.1 Events

Asynchrony plays an important role in computational systems. The need for asynchrony comes from our need to have interaction between modules of a system [63, 105]. To cope with asynchrony, programmers have described the event-driven programming model. Luckham [76] defines an event as an object that is a record of an activity in a system. Events are related to other events by time, causality, and aggregation. $E_1 \to E_2$ means that event E_1 caused E_2, i.e. E_1 had to happen before E_2, or, equivalently, E_2 depends on E_1.

An event is something that happens. Events are happening all around us all the time. They include a price change in a betting exchange market is, a transaction, a goal being scored and a key stroke. An event can be also something that does not happen, but was supposed to, e.g. a smoke detector that did not activate the horn after smoke detection. Each event is received and then processed according to the type of the event and the data that carries with it. Events can be produced almost anywhere in a system. The user, an I/O module, another computation or even the same computation can produce events. Some of the most famous examples using the event-driven model are: User Interfaces, Web Servers and Operating Systems (OS).

Event-based programs are organized around the processing of events. When a program cannot complete an operation immediately because it has to wait for an event, it registers a callback function that will be invoked when the event occurs [56, 78].

An event can be either primitive (e.g. a serve in a tennis match) or composite (e.g. a rally, that is a sequence of shots within a point, in a tennis match). Primitive events occur at a point in time (e.g. time of serve) while composite events occur over an interval (e.g. the interval starts at the time the rally started and ends when the point is won by one of the players).

Adaikkalavan et al. [1] proved that when events are detected using detection-based semantics, where event occurrence and event detection is not differentiated, it leads to incorrect detection of events. Primitive events are predefined in the system and are detected at the time of occurrence. Composite event detection on the other hand involves two steps:

(A) : Checking the detection condition based on the operator semantics

(B) : Determining the time of detection

Several formalisms have been proposed as being suitable for the detection of composite events in the literature. Some of them are: Event Detection Graphs (EDGs) [27], Extended Finite State Automata [75, 52], Colored Petri Nets [49, 50, 51], and Event Algebra [50, 21, 22]. In addition, many libraries, programming languages, compilers etc. have been developed over the last years as tools for developers in order to handle event driven programming. There are two ways to handle events in an application:

- Using the principles of thread programming.
- Creating a context for each event and processing it along with other contexts in one computation stream.

There has long been a debate about whether threads or events are best suited to systems software. The central question of this debate is whether threads or events should be used to manage concurrent I/O [33]. Threads can increase the execution efficiency of a program and can take advantage of multicore hardware. However, programming with threads can be difficult and as a result many programmers produce buggy software. On the other hand, event-based programming provides all the benefits that threads provide, but in a more robust way. Thread-based programs use multiple threads of control within a single program in a single address space. In threaded programs the programmer is responsible for the thread creation, synchronisation, termination, scheduling, process interaction and protection of shared data structures.

2.4.2 Event Stream Processing

An **event stream** is a linearly ordered sequence of events. Usually, streams are ordered by time e.g. arrival time such as in a betting exchange or stock market feed. An event stream may be bounded by a certain time interval or other criteria (content, space), or be open ended and unbounded [77]. A stream may contain events of many different types. **Event stream processing (ESP)** [39, 107] is a set of technologies designed to acquire and process multiple event streams in order to identify and react to meaningful, important or critical events within those streams.

ESP enables applications such as algorithmic trading in financial services, Radio-Frequency Identification (RFID) event processing applications [128, 120], fraud detection [38, 60], process and supply chain monitoring, and location-based services in telecommunications.

2.4.3 Complex Event Processing

Complex event processing (CEP) [32, 26, 2] is a technique that allows applications to analyse event data (multiple streams of events) in real-time, in order to find meaningful events within the event cloud and enable instant response to these changing conditions.

Complex Event Processing allows for real-time response. Latency is delay, and for some companies milliseconds of delay can mean millions of dollars in lost revenue. Greg Linden has claimed that every 100ms of delay experienced on the Amazon.com retail website reduced revenues by 1% [121]. Traditional data analysis tools are based on databases: the analysis logic is applied to static sets of data to extract the information that is needed to make decisions. By contrast, most CEP engines are event-driven: the analysis to be done is applied in advance, and each new event is processed as soon as it arrives, immediately updating all high level information and triggering any rules that might be defined.

A complex event processing algorithm can for example, contain the following rule:

"Alert a trader if a goal is scored in a football match x, which is followed within two minutes by a fall or a rise of greater than 5% in the value of the odds in any of the available selections in the Match Odds market."

2.4.4 Event-driven Programming Languages

Lua/ALua

Lua [115] is a powerful, fast, lightweight, embeddable scripting language created in 1993 at PUC-Rio, the Pontifical Catholic University of Rio de Janeiro in Brazil. Lua is a dynamic programming language and supports object-oriented programming, functional programming, and data-driven programming. Practically, Lua is realised as a library written in C.

Lua has been used in many industrial applications (e.g. Adobe's Photoshop Lightroom), with an emphasis on embedded systems (e.g. the Ginga middleware for digital TV in Brazil) and games (e.g. World of Warcraft)[4]. Lua is currently the leading scripting language in games.

ALua [115, 114] (Active Lua) is an event-driven communication mechanism for developing distributed parallel applications based on Lua. In ALua, a program is composed of processes called agents, running on one or more physical machines. These processes intercommunicate

[4]http://www.lua.org/about.html

through an asynchronous sending function. Each agent contains a Lua interpreter and an event loop, which manages network and user-interface events. Figure 2.10 shows the structure of an agent. Each event contains a piece of Lua code. The event loop continually receives events, and sends its contents to the Lua interpreter for immediate execution.

The user interface is a console, where the user can enter Lua commands. Each line the user types generates an event. Network events correspond to messages received from other processes. The sending function sends a piece of Lua code to be executed in another process. ALua has no explicit operation for receiving a message. The receiver's event loop will automatically execute the received code. With this mechanism, it is very easy to implement several typical distributed tasks, such as to call remote procedures or to inspect and modify remote variables [115].

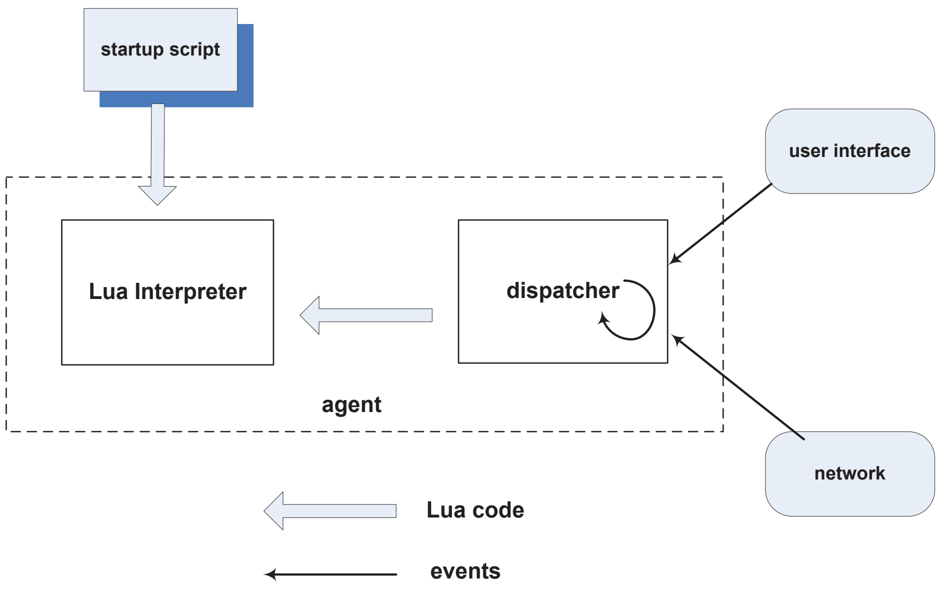


Figure 2.10: The structure of an ALua agent

UrbiScript

UrbiScript[5] [108] is a programming language primarily designed for robotics. According to UrbiScript's documentation "UrbiScript is a dynamic, prototype-based, object-oriented scripting language. It supports and emphasizes parallel and event-based programming, which are very popular paradigms in robotics, by providing core primitives and language constructs".

[5]See http://www.gostai.com/support/documentation

UrbiScript was initially developed in 2003 by Jean-Christophe Baillie in the Cognitive Robotics Lab of ENSTA, Paris. It is now actively and further developed in the industry through the Gostai Company founded in 2006. The core of Gostai's technology is the URBI middleware for robotics, which was initially developed for use with the AIBO series of robotic pets [9, 8, 85, 10, 108]. The URBI middleware comes with dedicated abstractions to handle event-based programming and concurrency from within C++, Java, Python or Matlab, together with a distributed component architecture called UObject which can be interfaced with Microsoft Robotics Studio and CORBA. URBI is based on a client/server architecture and a dynamic language called UrbiScript that can be used to coordinate the UObject components. URBI is the corresponding virtual machine used to run URBI scripts. It is conceptually similar to the Java Virtual Machine, or to a Python interpreter. The main difference with many current interpreted languages is that URBI allows you to build your own virtual machine, enhance it with plugins and make other applications to converse with it. The innovation of UrbiScript as a programming language is that it brings new abstractions to handle parallelism and event-based programming, directly integrated into the language semantics. UrbiScript programs routinely run hundreds of parallel threads and react to several events at the same time. The idea is to separate the logic of the program on one side (UrbiScript) and the fast algorithms on the other side (UObjects in C++, Java, Python etc.), and use the dynamic capabilities of scripting languages to build the glue. This is already a widely-used approach in video games, with scripting languages like Lua (described previously), or Python. URBI brings this same approach to robotics and improves the scripting language side with parallelism and event-based programming (see Figure 2.11).

Before we continue further it is necessary to introduce some specialist vocabulary relating to URBI:

URBI Kernel : the URBI kernel is the heart of URBI. It is the library holding all the core features needed to execute user scripts and requests.

URBI Cores : an URBI core is a kernel linked to system-specific functions needed to run URBI.

URBI Engine : An URBI engine (or URBI server) is a full program, on which scripts can be executed. This server can be executed directly on a robot or a computer, and can receive connections from clients. An URBI Engine is practically realised as an URBI core linked with any number of UObjects.

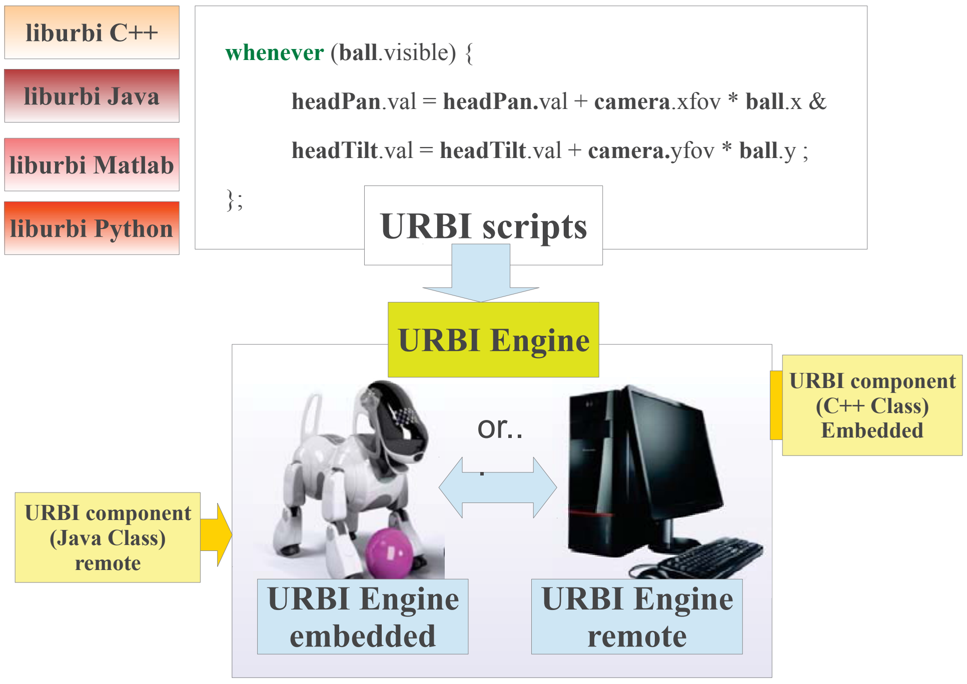
liburbi : This is a C++, Java, Pyhton or Matlab library, used to allow any program to send commands to an URBI server, and to receive its responses. It allows the integration of URBI into third party applications.

UObjects : a UObject is a plugin, which can be a C++, Java, Python or Matlab class. UObjects are usable by URBI, either as a component of a server (a plugin UObject), or as a standalone application connected to an URBI server (a remote UObject). Only C++ classes can be plugin UObjects while other types of UObjects must be remote UObjects.

Connection : An URBI server can accept incoming connections from any computer. A connection designates a channel through which a client (an application or an interactive user shell) sends commands to URBI and receives responses. An URBI server can handle an arbitrary number of network connections from local or remote clients.

Figure 2.11 shows the way URBI components interact with each other. The URBI Engine is the heart of the system. It can execute on a robot (embedded) or on a computer (remote). In case of a remote engine, the engine is responsible for forwarding requests and answers to and from the robot. The standard URBI Engine can be augmented with plugins. Once the URBI Engine is running, it is possible to connect to it, and to send commands. This is done with a user shell or with any application using liburbi to send and receive messages.

As we saw previously the most current mainstream programming languages rely on threads or complex abstraction classes to handle parallelism and the complexity that comes with it, like concurrent access to shared resources. Events are usually handled by a custom-made synchronous event loop with corresponding event queues. On the other hand what makes UrbiScript different is that it integrates parallelism and events in the core of the language semantics.



```
whenever (ball.visible) {
        headPan.val = headPan.val + camera.xfov * ball.x &
        headTilt.val = headTilt.val + camera.yfov * ball.y ;
};
```

Figure 2.11: URBI Architecture

As previously discussed, any C++, Java, Python or Matlab class can inherit UObject and become visible inside UrbiScript as a regular object that can interact with other objects already plugged in. UObjects can be either linked to the URBI Engine and share its memory and other UObjects memory, or it can also be run remotely as an autonomous separate process. In this case, the interface between URBI and the UObject is handled transparently through the network via TCP/IP and the UObject is reflected inside the UrbiScript language in exactly the same way as in the “linked” mode. This allows having large distributed network of objects interconnected through scripts coordinating them in a parallel and event-driven way, which contrasts with the traditional one-to-one component interactions of similar component architectures [8, 85]. In Figure 2.12 we can see how URBI is capable of automatically load balancing parallel commands on a variable number of cores and threads in real-time.

UrbiScript brings a number of powerful abstractions that are very hard to access in other scripting languages. For example, in UrbiScript you can request any variable to reach a value over a given time or at a given speed. Also any portion of code can be prefixed with a tag. You can then later stop, freeze, unfreeze this code from anywhere using the tag name.

General event processing models receive inputs (mainly in the form of events) from the external

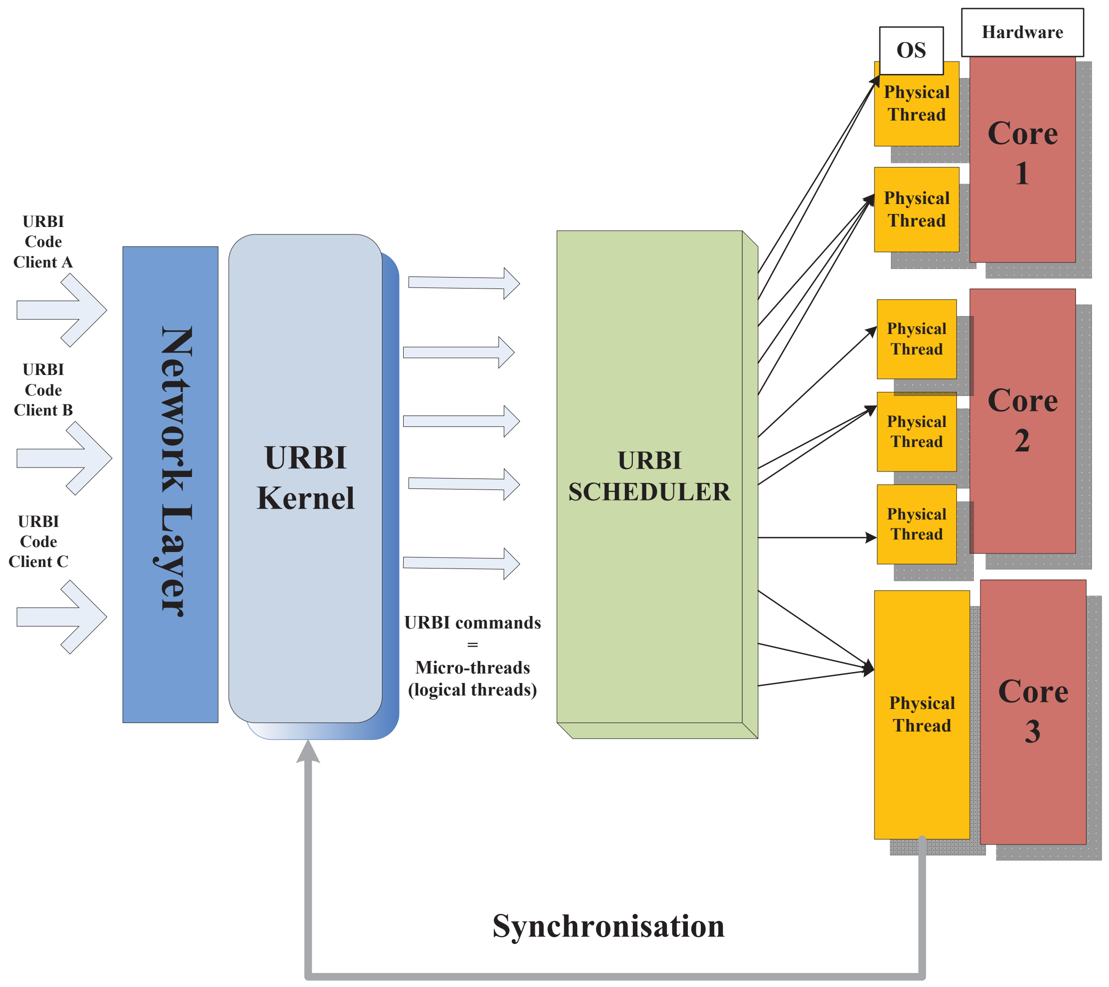


Figure 2.12: URBI multicore integration [7]

environment and react by performing actions that change the stored information (internal actions) or influence the environment itself (external actions). Once the events are detected, they are used to trigger a set of **Event Condition Action (ECA)** rules. ECA languages have reactive rules of the form **On** *Event* **If** *Condition* **Do** *Action* which mean: when Event occurs, if Condition is verified, then execute Action. There are many potential and existing areas of applications for ECA languages such as Active and Distributed Database Systems [49, 1, 28], Semantic Web applications [84, 73] and Distributed Systems [55, 87].

Clearly ECA languages deal with systems that evolve, such as a betting exchange system. To the best of our knowledge no existing ECA language or Event-Driven programming language exists for betting exchange markets. As nothing similar exists so far, the present research can be the basis for further research and development in this area; moreover, it can be insightful and suggestive of new perspectives in other related areas.

2.5 Stochastic Optimisation

Optimisation is a branch of applied mathematics and numerical analysis that deals with the optimisation of a function or a set of functions according to some criteria. In many nonlinear optimisation problems, the objective function f has a large number of local minima and maxima. Finding an arbitrary local optimum is relatively straightforward by using classical local optimisation methods [101, 66], but finding the global minimum (or maximum) of a function is often a very hard challenge. Decision makers often have to make decisions in the presence of uncertainty. Decision problems are often formulated as optimisation problems, and thus in many situations decision makers wish to solve optimisation problems which depend on parameters which are unknown [6].

Stochastic optimisation algorithms have been growing rapidly in popularity over the last two decades, with a number of methods now becoming "industry standard" approaches for solving challenging optimisation problems. These are algorithms used to find answers to problems when (a) one does not know what the optimal solution looks like, (b) there is no known rigorous method for finding it in a principled way, (c) exhaustive search is out of the question because the space is too large, but (d) it is possible to efficently test and assess the quality of candidate solutions [43, 90, 91, 37].

To optimise a trading strategy is to obtain its peak trading performance. Most trading strategies have a set of parameters that highly affect their performance. Any strategy that can accept different values for these parameters is eligible for optimisation. An optimisation algorithm is an algorithmic method that can be applied to solve optimisation problems. Numerous optimisation algorithms are available but choosing one for solving a given optimisation problem depends much on the characteristics of the optimisation problem at hand [95]. Many optimisation methods are especially designed for specific types of search spaces, objective and constraint functions. This work focuses on optimisation methods that are not dependent on any knowledge about the system or model of the optimisation problem. For example: imagine that you are trying to find an optimal set of parameters for a betting exchange strategy. You have a simulator for the betting strategy and can test any given set of parameters and assign it a quality. And you have come up with a definition for what a strategy parameter sets looks like in general. But you have no idea what the optimal parameter set is, not even how to go about finding it. These optimisation problems are known as black box optimisation problems [127] (see Figure 2.13)

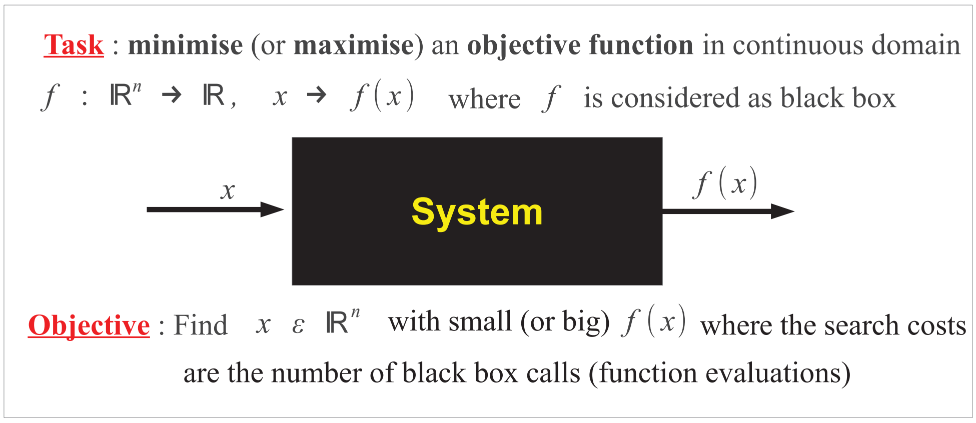

Task : minimise (or maximise) an objective function in continuous domain
$f : \mathbb{R}^n \rightarrow \mathbb{R}, \quad x \rightarrow f(x)$ where f is considered as black box
x
System
$f(x)$
Objective : Find $x \in \mathbb{R}^n$ with small (or big) $f(x)$ where the search costs are the number of black box calls (function evaluations)

Figure 2.13: Black box optimisation

2.5.1 Overfitting

Optimisation has many pitfalls and so there are many ways that it can be done incorrectly. A primary pitfall is that of overfitting [99, 122]. This occurs when the optimisation process identifies parameters that produce good trading performance on historical data but produce poor trading performance on unseen data. This is because someone can always find a combination of rules and trading parameters that fits perfectly to the available historical data, resulting in exceptional trading results based on those tests. However, when those rules are tested on a live market, they can fail and lose money very quickly. There are degrees of overfitting. A trading strategy, where the degree of overfitting is not extreme, can still produce real-time profit but it will certainly under-perform its historical results. On the other hand, a highly overfitted trading strategy will produce disastrous real-time trading losses. Thus to avoid overfitting, it is essential to further test any strategy with the optimised parameters on a set of historical data that is distinct from that used in the optimisation process. Some of the most well known and used techniques are the k-fold Cross Validation [96, 4, 67], Regularisation [54, 3] and Walk-Forward Analysis [93]. In SPORTSBET optimisation platform Walk Forward Analysis is employed to avoid overfitting (see Chapter 5).

2.5.2 Cross-Validation

Cross-validation is primarily a way of measuring the predictive performance of a statistical model.

The most common types of cross-validation are:

- **K-fold cross-validation**: First, a random permutation of the original sample set is generated and randomly partitioned into k subsets ("folds") of about equal size. Of the k subsamples, a single subsample is retained as the validation data for testing the model, and the remaining $k-1$ subsamples are used as training data. The cross-validation process is then repeated k times (the folds), with each of the k subsamples used exactly once as the validation data. The k results from the folds then can be averaged (or otherwise combined) to produce a single estimation [69, 97].

- **Repeated random sub-sampling validation**: This method randomly splits the dataset into training and validation data. For each such split, the model is fit to the training data, and predictive accuracy is assessed using the validation data. The results are then averaged over the splits. The advantage of this method (over k-fold cross validation) is that the proportion of the training/validation split is not dependent on the number of iterations ("folds"). The disadvantage of this method is that some observations may never be selected in the validation subsample, whereas others may be selected more than once. In other words, validation subsets may overlap [69, 15].

- **Leave-one-out Cross-Validation**: involves using a single sample from the original sample set as the validation data, and the remaining samples as the training data. This is repeated such that each sample in the sample set is used exactly once as the validation data. This is the same as k-fold cross-validation where k is equal to the number of samples in the sample set [69, 24].

Chapter 3

UBEL: a Universal Betting Exchange Language

This chapter presents a novel generic betting strategy specification language called UBEL. UBEL is based on extensions to the UrbiScript programming language and aims to allow its users to focus on trading ideas rather than low-level programming issues.

We begin by giving a high-level perspective of UBEL's architecture showing how it integrates with Urbiscript. After motivating the need for a new domain-specific language, we recap the main features of UrbiScript and highlight the areas in which it can be enhanced. We proceed to show how UrbiScript has been adapted to fit the domain of betting exchange strategies. Finally, UBEL is described in detail. Examples of actual code are included to give a flavour of the language.

3.1 UBEL Architecture

Figure 3.1 shows the high-level architecture of UBEL. There are five main components:

1. **UrbiScript core** providing math functions, flow control constructs, parallelism and concurrent flow control, event-based programming features etc.

2. **Generic betting classes** which are reflect essential components and features of a betting exchange system.

3. **Sport-specific betting classes** which reflect specialist aspects of sports such as Tennis, Football, Horse Racing etc. Users can define their own classes by inheriting from the generic class `SportingEvent`.

4. **UObjects** which are external modules written in another language (Java, C++, Python, etc.). These are treated in the same way as native UBEL objects using the UObject API. This allows, for example, to connect a hierarchical Markov chain model for tennis written in C++ to be very easily connected with the UBEL world.

5. **Betting functions** for common advanced trading operations such as dutching, spreading profit across markets etc.

3.2 UBEL Motivation and Benefits

3.2.1 The Need for a DSL

A Domain Specific Language (DSL) is a concise programming language designed specifically to naturally express solutions to problems in a specific domain [116]. Examples of commonly-used DSLs are LaTeX for writing academic papers, Matlab for numerical linear algebra algorithms and SQL for querying relational databases.

A well-designed DSL can be much easier to program and use with than a traditional programming language, because it provides a concise notation exploiting domain-expert vocabulary and abstractions. This improves programmer productivity. Moreover, DSLs may also improve the communication of programmers with domain experts, which is often a major bottleneck in software development.

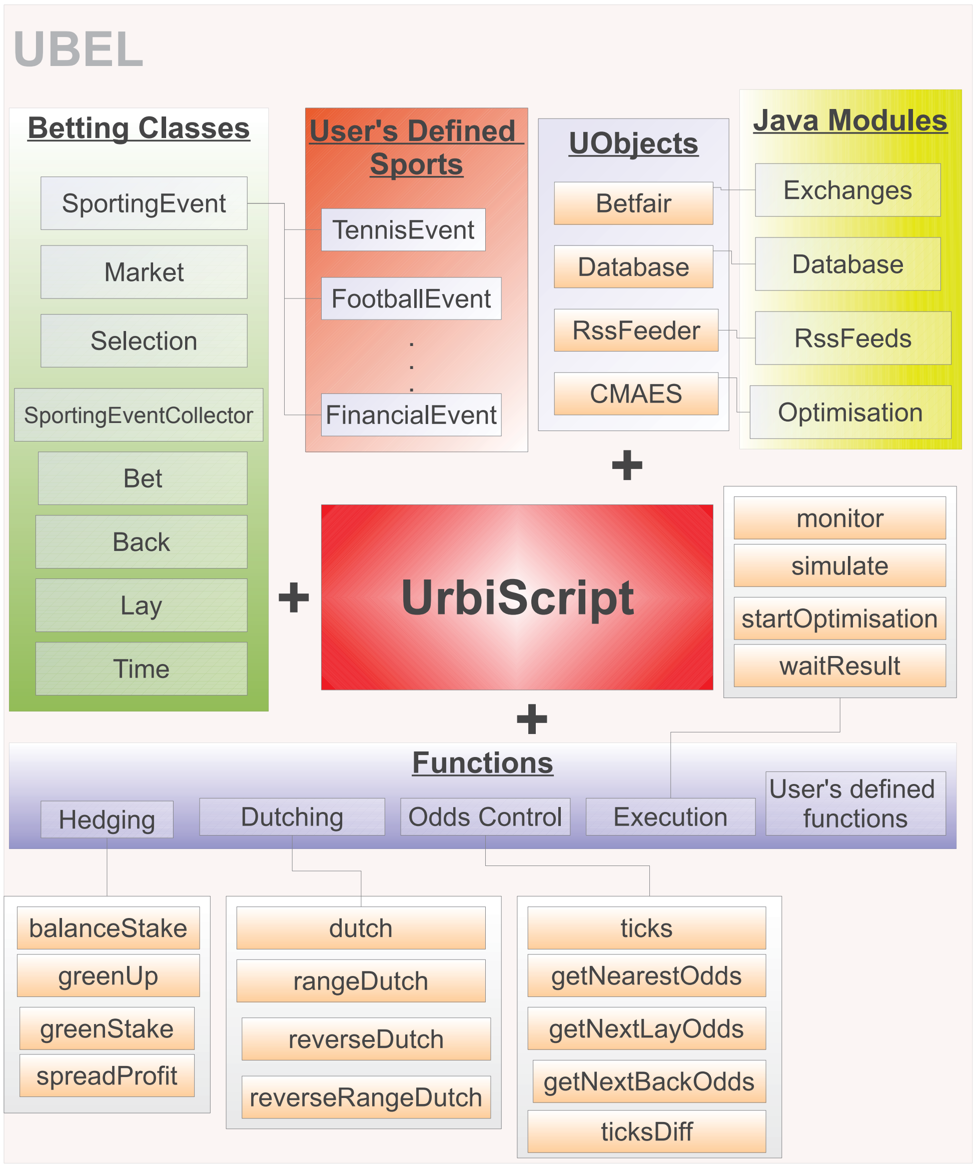


Figure 3.1: High level architecture of UBEL

3.2.2 DSL Approaches

DSLs come in two main forms:

External DSL: An external DSL is a language that is parsed independently of the host general purpose language. There are two major challenges with this approach. First, it needs a huge effort to develop a new language to a sufficient degree of maturity. This effort would have to include things such as: language specification, optimised compiler, libraries, a modern IDE, debugging tools etc. Second, DSLs sometimes have to interface to other parts of a system. It is not clear how multiple separate DSLs would be used to work together without creating a new DSL that integrates the desired combination of the others [25].

Internal DSL: An internal DSL lives inside of a host language. It is quite like a framework or a library for the host language, and typically consists of a set of classes and operations on objects of those types. There are many advantages to using an internal DSL. First, programmers do not have to learn a completely new syntax. Second, multiple DSLs can be combined in the same application without need to create a new DSL. Third, all the facilities and components of the host language (such as compilers, debuggers etc.) can be readily reused. However, the main challenge with internal DSLs is that they cannot exploit domain knowledge to efficiently map programs to specialized architectures [25].

We have developed UBEL as an internal DSL using URBI as the host framework. By doing so, we produced a more powerful language than other methods since many features of URBI come with it. The development effort was modest because the existing implementation can be reused, the host language infrastructure can be reused which includes the development and debugging environments and since the language has similar syntax with well known object oriented and functional programming languages the effort of the user to learn the language will be minimum.

3.2.3 Comparison with XML

Assume we have this very simple strategy example: "*When a market turns in-play, place a lay bet on any selection whose price drops below a certain limit but not if it is going to be the first bet and the selection's price is less than 1.1*". Before the introduction of UBEL one might think to specify this strategy using XML, a standard open markup language for encoding documents in a format that is readable by both humans and machines. However as XML does not natively support any sort of betting terminology, someone should have to design first a Document Type Definition (DTD) document which defines a set of statements that follow the rules of the XML and are specific for this betting strategy. The result of that would be just a specification of a strategy and nothing else. In Listing 1 we can see the strategy written as an XML file [41].

The definition of the strategy using XML is very cumbersome and difficult to understand and things become even more difficult when concurrency must be included in a strategy. Moreover, each time you want to use the same strategy with different parameters you need to edit the XML file.

```
<XML>
  <block ID="11871883510" Active="1" Name="Laybelow limit" Repeat="0.00" RepeatTimes="0" Protected="0">
    <Triggers>
      <trigger>
        <BlockID>11871883510</BlockID>
        <Action>1</Action>
        <Active>2</Active>
        <CondMet>0</CondMet>
        <Market>2</Market>
        <Selection>4</Selection>
        <Persistence>0</Persistence>
        <Price>lay_price</Price>
        <Amount>4/lay_price</Amount>
        <MarketStatus>1</MarketStatus>
        <Name>laying on selections below 2.0</Name>
        <RepeatTimes>0</RepeatTimes>
        <TriggerName/>
        <AdjustAmounts>0</AdjustAmounts>
        <WinLose>0</WinLose>
        <UseOptions>0</UseOptions>
        <OptNoBet>0</OptNoBet>
        <OptPersistence>0</OptPersistence>
        <OptPersistenceValue>0</OptPersistenceValue>
        <OptAmount1>0.01</OptAmount1>
        <OptAmount2>10000000</OptAmount2>
        <OptPrice1>1.01</OptPrice1>
        <OptPrice2>1000</OptPrice2>
        <Repeat>0.00</Repeat>
        <conditions>
          <tblck AndOr="0">
           <blcks>
            <tblck AndOr="0">
             <blcks/>
              <conds>
               <condition AndOr="1" Body="0" Param="1" Condo="2" Val1="2.0" Val2="" TrName="" Time="0" Frml=""/>
               <condition AndOr="1" Body="13" Param="7" Condo="3" Val1="0" Val2="" TrName="" Time="0" Frml=""/>
              </conds>
             </tblck>
            </blcks>
              <conds>
            <condition AndOr="0" Body="0" Param="1" Condo="6" Val1="1.1" Val2="1.99" TrName="" Time="0" Frml=""/>
              </conds>
          </tblck>
        </conditions>
      </trigger>
     </Triggers>
    </block>
</XML>
```

Listing 1: Strategy example in XML

Moreover, XML and its extensions have regularly been criticised for verbosity and complexity. XML is not very practical if you expect human beings to author the code by hand. In addition, mapping the basic tree model of XML to type systems of programming languages or databases can be difficult, especially when XML is used for exchanging highly structured data between applications, which was not its primary design goal [125].

In Listing 2 we can see the same strategy written in UBEL. The strategy is written as a function which takes the strategy parameters as inputs. Likewise, that the strategy is more clear and more flexible. UBEL simplifies the process of writing both simple and complex strategies for betting exchange markets.

```
function layBelowLimit(var sportingEvent, var limit, var payout){

  var numberOfBets = 0;

  for&(var market:sportingEvent.markets){

    at(market.inplay?){

      for&(var selection:market.selections){

        var selectionTag = Tag.new;

        selectionTag:{

          at(selection.updated?){

            var odds = selection.lp[0];

            if(odds<limit && !(numberOfBets ==0 && odds<=1.1)){

              var stake = payout/odds;
              var myBet = Lay.new(market.name,selection.name,selection.lp[0],stake);
              sportingEvent.placeBet([myBet]);
              numberOfBets = numberOfBets+1;
              selectionTag.stop;

            };
          };
        };
      };
    };
  };
};
```

Listing 2: Strategy example in UBEL

3.3 UrbiScript Basic Features

- **Functions**

  UrbiScript provides built in functions with the classical, mathematical notation.

- **Variables**

  Variables can be introduced with the *var* keyword, given a name and an initial value [1].

[1]See http://www.gostai.com/support/documentation

Table 3.1: UrbiScript Variable Types

Description	**Type**
var i = 5.0;	Float
var myList = [1,2,"three"];	List
var myStr = "hello"; myStr = myStr + "world";	String
var myMap = [key ⇒ value];	Dictionary
nil	Neutral value
void	Absence of value

- **Method calls** : Methods are called on objects with the dot (.) notation as in C++. Method calls can be chained.

- **Function definition** : Functions can be declared using the *function* keyword, followed by the name of the function, parentheses surrounded list of formal arguments, and the body between curly brackets.

- **Objects** : An object in UrbiScript is a list of slots. A slot is a value associated to a name. So an object is a list of slot names, each of which indexes a value just like a dictionary. In UrbiScript everything is an object.

- **Methods** : Methods in UrbiScript are simply object slots containing functions.

- **Parallelism, Concurrent Flow Control** : Parallelism is one of the major features of UrbiScript. UrbiScript provides you with some very powerful time operators (see Table 3.2).

Table 3.2: UrbiScript Time Operators

UrbiScript Command	Description	Comment
A ; B	“ ; ” Serialization operator. execute first A, then B	Wait for the A to finish before continuing B (no time Constraint)
A , B	“ , ” Background operator. A is started, and then immediately B	B starts as soon as it is possible after A (A can be not fully ended)
A & B	“ & ” Parallelism operator. A and B start simultaneously, and executed in parallel	This command returns when A AND B are ended
A | B	“ | ” B starts immediately after A	There is no delay between the end of A and B
{A ; B , C & {D | E }};	group commands is available with “{” and “}”	Useful for complicated tasks

Example: Assume that we have the function below:

Function definition
```
function print(name){

echo(name + ": 1");

echo(name + ": 2");

echo(name + ": 3");

};
```

This function take as input one variable called *name* and prints out three times the name of the variable following by a number. So if you want to have serialized executions you can write:

Serialized Execution
```
print("left"); print("middle"); print("right");
```

For parallel execution you can write:

Parallel Execution
```
print("left") & print("middle") & print("right");
```

and the outputs would be:

Serialized

```
*** left: 1
*** left: 2
*** left: 3
*** middle: 1
*** middle: 2
*** middle: 3
*** right: 1
*** right: 2
*** right: 3
```

Parallel

```
*** left: 1
*** middle: 1
*** right: 1
*** left: 2
*** middle: 2
*** right: 2
*** left: 3
*** middle: 3
*** right: 3
```

The difference between the parallelism and background operator is that the execution of a code is blocked after a " & " group until all its children have finished. On the other hand the execution of a code after a " , " will be run immediately.

One important thing to notice is that UrbiScript allows you to use parallelism operators in loops. What that means is that you can execute all the commands inside a loop in parallel.

Parallelism operator in for loop

```
for&(i=0 ; i< 10 ; i++) {
//commands that will be executed in parallel
};
```

These operators are really powerful and enable you to include parallelism anywhere at no syntactical cost.

- **Groups** : UrbiScript allows you to group objects in "groups" with the group command and send messages to all of them as if they were one.
- **Flow Control Constructs** : The flow control constructs (*if, while, for, do, switch, foreach*) are very common across many programming languages, and they are used to perform different computations or actions depending on whether a specified boolean condition evaluates to true or false. UrbiScript provides that and for practical reasons, UrbiScript has added two more constructs, *loop* and *loopn* to create infinite loops in the first case and loops iterating n times in the second case.
- **Tags for parallel control flows** : UrbiScript allows you to tag a block of code. One of the primary purposes of tags is to be able to control the execution of code running in parallel. This is possible because tags have a few control methods such as:

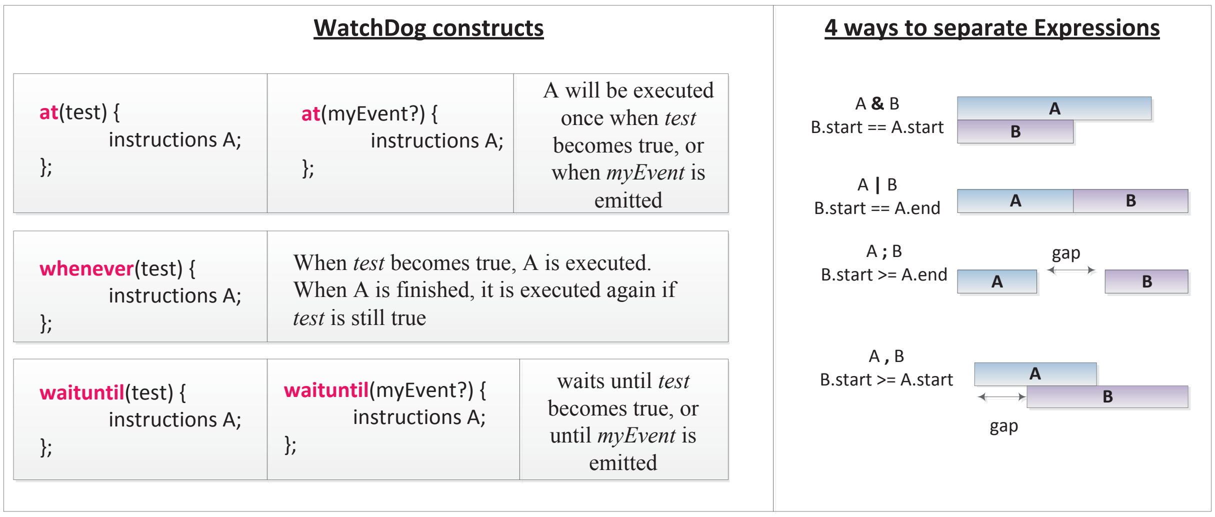
  - *freeze* : Suspend execution of all tagged code.
  - *unfreeze* : Resume execution of previously frozen code.
  - *stop* : Stop the execution of the tagged code.
  - *block* : Block the execution of the tagged code.
  - *unblock* : Stop blocking the tagged code.

- **Event-based Programming** : UrbiScript has a strong support for event-based programming. It provides you with event related constructs such as:

  - *at* : Given a condition, and an expression, *at* will evaluate the expression every time the condition becomes true. *at* works a bit like *if*, except that is it always running in the background.
  - *onleave* : Block can be appended to *at* to execute an expression when the expression becomes false. *onleave* is a bit like *else* and is followed by an action that will be executed when the condition (test) switches from true to false.
  - *whenever* : construct is similar to *at*, except that the expression evaluation is systematically restarted when it finishes as long as the condition stands true. *whenever* works a bit like while, except that it never terminates and run in the background.
  - *else* : Block can be appended to whenever to execute an expression as long as the condition is false.
  - *wait* : The command *wait(n)* will wait for *n* milliseconds before ending.
  - *waituntil* : The command *waituntil(condition)* waits until the condition becomes true and can be useful to synchronize different parallel programs on a given condition.
  - *timeout* : The command *timeout(n) "something"* will execute the command *"something"* and stop it after *n* milliseconds if it is not already finished.
  - *stopif* : The command *stopif(condition) "something"* will execute the command *"something"* and stop it when the condition becomes true. If the command is already finished, nothing happens.

  - *freezeif* : The command *freezeif(condition) "something"* will execute the command *"something"* and freeze it when the condition becomes true. When the condition is false again, *"something"* unfreezes.

- **Events** : UrbiScript allows you to define events, that can be caught with some of the constructs we saw above like the *at* and *whenever*. So you can actually create events by cloning the Event object and you can then emit them using the *" ! "* keyword. You can also add parameters to events and when the event is caught, you can retrieve the parameters.



WatchDog constructs

`at(test) { instructions A; };`	`at(myEvent?) { instructions A; };`	A will be executed once when *test* becomes true, or when *myEvent* is emitted
`whenever(test) { instructions A; };`	When *test* becomes true, A is executed. When A is finished, it is executed again if *test* is still true	
`waituntil(test) { instructions A; };`	`waituntil(myEvent?) { instructions A; };`	waits until *test* becomes true, or until *myEvent* is emitted

Figure 3.2: UrbiScript's main features

3.4 UBEL Implementation

The first target of this research was to create an novel generic betting specification language which captures the notion of parallelism and event-based programming. UrbiScript provided everything we needed except from the betting classes and the processes which model the behaviour of a betting exchange system.

Before we continue with the implementation we will present you some terminology regarding the betting exchange markets and UBEL, to avoid confusion.

3.4.1 Terminology

- **Sporting Event** : Can be a football match, a horse race, a tennis match etc. Examples:
  - "*Real Madrid v Liverpool* " (football match)
  - "*Beverley Minster Claiming Stakes (Class 5) — 5f*" (horse racing)
  - "*Federer v Nadal*" (tennis match)
- **Event** : An event as mentioned in previous Chapters is something that happens. Some examples of events are:
  - an increase or decrease of the odds in one of the *Selections* in a *Market*.
  - an increase or decrease in the volume of money available for betting, in one of the *Selections* in a *Market*.
  - a red card in a football match.
  - a goal being scored in a football match.
  - a match point in a tennis match.

  All the above event examples are primitive. However, an event can be composite, so actually a combination of primitives. Example:

  - a goal being scored in a football match and the odds in one of the *Selections* in the *Market* decrease 10% .

3.4.2 UBEL classes

Urbi allows the use of remote components as if they were local, allows concurrent execution, makes synchronous or asynchronous requests and so forth. The UObject architecture provides a common API that allows components to be used seamlessly in highly concurrent settings. Components with an UObject interface are naturally supported by the UrbiScript programming language. Like that someone can interact with these components by making queries, changing them, observing their state, monitoring various kinds of events and so forth.

For the implementation of UBEL we decided to combine UObjects and pure UrbiScript classes. The UObjects are simply being used to interact with external resources while the pure UrbiScript classes are mainly modelling a betting exchange system. In Figure 3.3 we can see a UML diagram of the pure UrbiScript classes of UBEL, while in Figure 3.4 the UObjects.

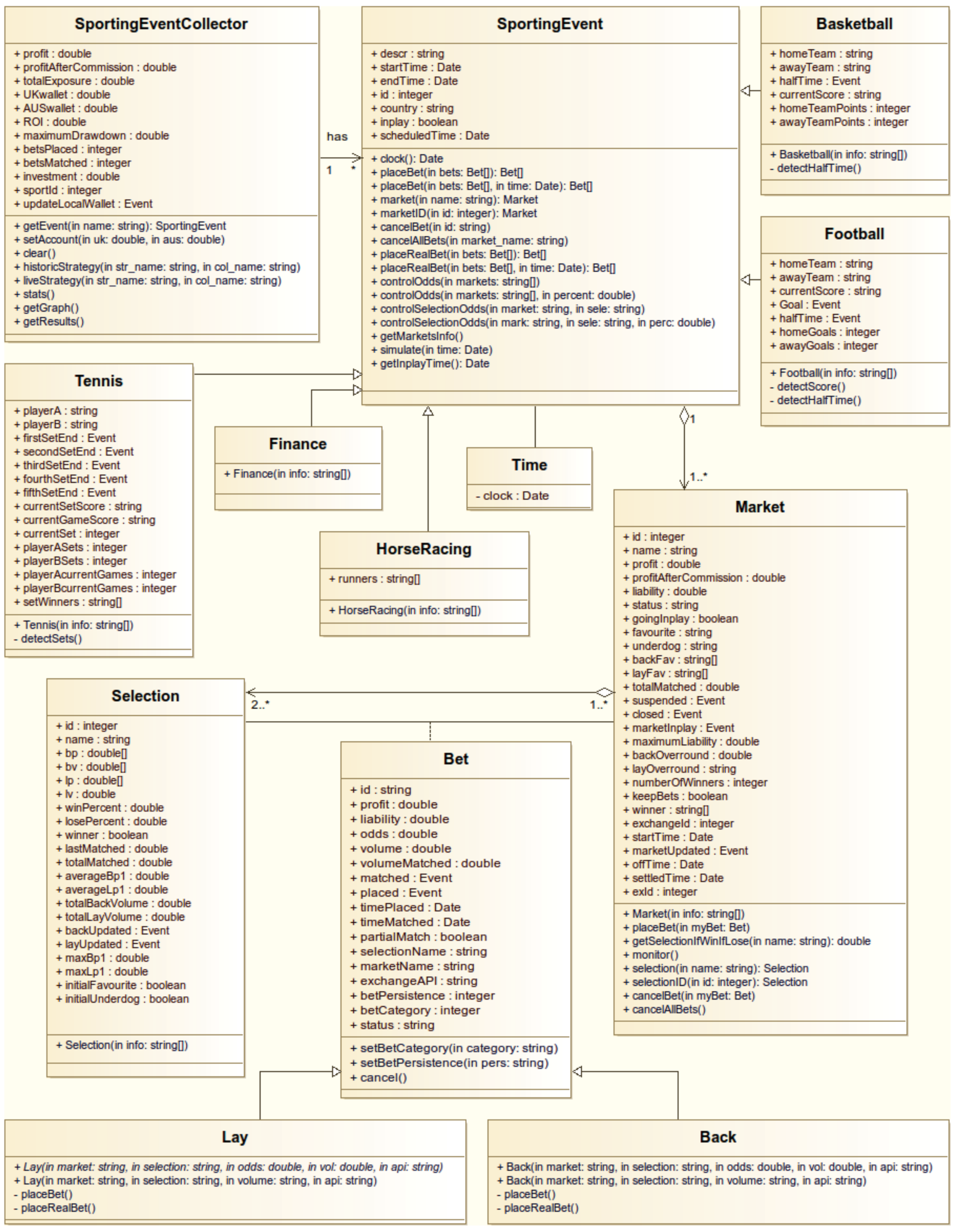


Figure 3.3: UBEL pure UrbiScript classes diagram

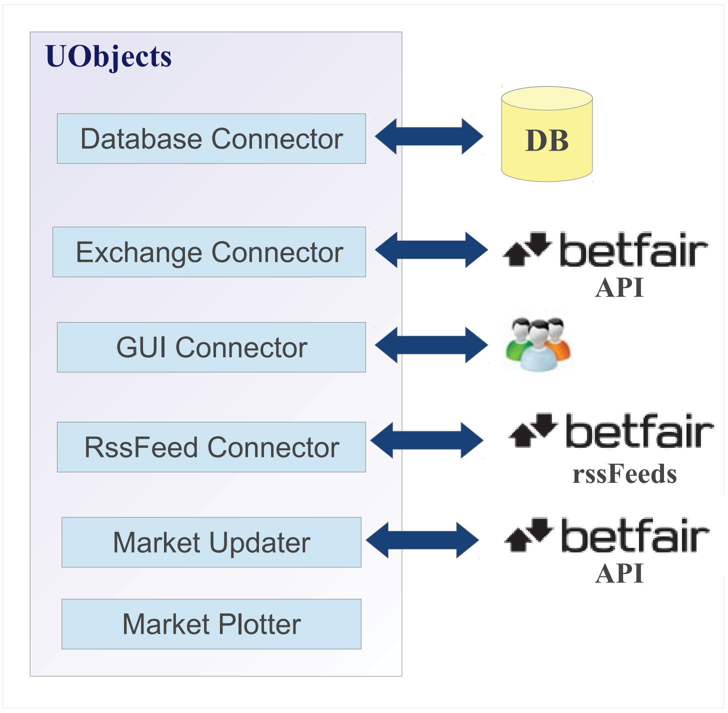


Figure 3.4: UBEL UObjects

SportingEventCollector

The highest level class is the `SportingEventCollector`. Every strategy includes a set of sporting events (e.g. football, tennis matches, horse races, etc.), on which bets can be placed. A `SportingEventCollector` object is responsible to find (in real time, or from a database) this set of sporting events, taking into consideration the specification by the user.

`SportingEventCollector` is the collector of all sports supported in SPORTSBET. At the moment SPORTSBET support tennis, football, basketball, horse racing and finance markets. Moreover, a `SportingEventCollector` object holds statistics for the strategy being associated with, such as Profit, Profit after Commission, Return on Investment and Maximum Drawdown. The user can very easily access them any time using the corresponding attribute name.

Constructor

```
function init(var sportId, var exchangeName){...};
```

Constructs a `SportingEventCollector` object for the specified sport and betting exchange platform.

Input Parameters :

- **sportId** : The id of the sport. (1 = Football, 2 = Tennis, 7 = Horse Racing, 7522 = Basketball, 6231 = Finance)

- **exchange** : The name of the betting exchange. Ex. “Betfair”.

```
Construction of Football Collector
var test = SportingEventCollector.new(1,"Betfair");

*** SportingEventCollector_0xfffffffdc3e6298
```

Main attributes

- **events** : A list of sporting events. Initially empty.

- **profit** : The difference between the winning and the losing trades.

- **profitAfterCommission** : The difference between the winning and the losing trades including commission.

- **account** : The account is the position you actually hold in a real-time trading account, whether it is a SPORTSBET internal simulation account (virtual money) or your live real-money betting exchange account.

- **numberOfBetsPlaced** : The total number of bets placed.

- **numberOfBetsMatched** : The total number of bets matched.

- **backBets** : The total number of back bets placed.

- **layBets** : The total number of lay bets placed.

- **backBetsMatched** : The total number of back bets matched.

- **layBetsMatched** : The total number of lay bets matched.

- **winnings** : The number of winning trades. A winning trade is the one where the net profit in a market is positive.

- **losses** : The number of losing trades. A losing trade is the one where the net profit in a market is negative.

- **maxWins** : The strategy's maximum number of consecutive wins.

- **maxLos** : The strategy's maximum number of consecutive losses.

- **maxDrawdown** : The Maximum Drawdown (MDD) measures the largest single drop from peak to bottom in an account balance during the life of a strategy. Formula:

  $$MDD = PV - LV \tag{3.1}$$

  where PV is the peak value before the largest drop and LV is the lowest value before new high established.

- **maxDrawdownPercent** : The Maximum Drawdown (MDD) in percent. Formula:

  $$MDD(\%) = \frac{MDD}{PV} \tag{3.2}$$

- **maximumRunUp** : The Maximum Run UP (MRU) measures the largest single increased from bottom to peak in an account balance during the life of a strategy. Formula:

  $$MRU = PV - LV \tag{3.3}$$

  where LV is the lowest account before the largest increased and PV is the highest value before new lowest established.

- **maximumRunUpPercent** : The Maximum Run UP (MRU) in percent. Formula:

  $$MRU(\%) = \frac{MRU}{LV} \tag{3.4}$$

- **ROI** : The Return On Investment evaluate the efficiency of an investment. Formula:

$$ROI(\%) = \frac{Profit\ after\ commission}{Total\ Investment} \tag{3.5}$$

- **RAR** : The Risk-Adjusted Rate of Return (RAR) measures the value of risk involved in an investment's return.

$$RAR(\%) = \frac{Net\ profit\ after\ commission}{Risk + Initial\ Account}, \quad where\ \ Risk = 2 \times MDD \tag{3.6}$$

- **RRR** : The Reward to Risk Ratio (RRR) provides an easy comparison of reward to risk.

$$RRR(\%) = \frac{Net\ profit\ after\ commission}{MDD} \tag{3.7}$$

- **perfectProfit** : The Perfect Profit is the total profit produced in case the strategy won the highest net profit in each market during the historical period.

- **PROM** : The Pessimistic return on margin (PROM) is a measure that pessimistically assumes that a trading strategy will win less and lose more in real-time trading than it did in its historical simulation. Formula:

$$PROM(\%) = \frac{[\overline{W} \times (WT - \sqrt{WT})] - [\overline{L} \times (LT - \sqrt{LT}]}{margin} \tag{3.8}$$

where $\overline{W}$ is the average winnings, $\overline{L}$ is the average losses, WT the number of winning trades, LT then number of losing trades, and margin the initial account balance. PROM is a robust measure because it takes into account a number of significant performance statistics as the ones mentioned above. Moreover, PROM penalises the small trade samples because of the adjustment of gross profit and loss by the square root of their respective number.

- **SE** : The Strategy Efficiency(SE) measures how efficiently a trading strategy converts the perfect potential profit into realised trading profits.

$$SE(\%) = \frac{Net\ profit\ after\ commission}{Perfect\ profit} \tag{3.9}$$

The user can very easily access all the attributes of a `SportingEventCollector` object by using their corresponding names.

Accessing SportingEventCollector attributes
```
test.profit;

*** 50.45

test.ROI;

*** 12.33
```

Main functions

A `SportingEventCollector` object can be assigned for historical simulation or real-time trading. This can be achieved by calling the functions: `historicStrategy` or `liveStrategy` and passing as parameters the name of the strategy and the variable name.

Historical and Live association
```
// Historical

var test = SportingEventCollector.new(1,"Betfair");

test.historicStrategy("name of the strategy","test");


// Live

var test2 = SportingEventCollector.new(2,"Betfair");

test2.liveStrategy("name of the strategy","test2");
```

- ***getFootballEvents***, ***getTennisEvents***, ***getBasketballEvents***, ***getFinancialEvents***, ***getHorseRacingEvents*** : request historical sporting events from the database. The user can ask for specific events and markets by passing as input the appropriate values.

  Input Parameters :

  - **selections** : The name of the selections. e.g. [“Milan”,“Inter”].
  - **excludeSelections** : The name of the selections to be excluded.

- **markets** : The name of the market. e.g. [“Match Odds”,“Set Betting”].
- **excludeMarkets** : The name of the markets to be excluded.
- **countries** : The name of the countries. e.g. [“Spain”,“GR”].
- **description** : Key words of a sporting event. e.g. [“Wimbledon”,“ATP”].
- **excludeDescription** : Key words of a sporting event to be excluded.
- **numberOfRunners** : The number of runners in a market. Apply only for Horse Racing. e.g. [5,10] will return markets were $5 \leq number\ of\ runners \leq 10$.
- **numberOfWinners** : The number of winners in a market. Apply only for Horse Racing. e.g. [1,1] will return markets were there is only one winner.
- **totalMatched** : The minimum amount of total matched money in a market.
- **exchangeId** : The id of the exchange (1 = UK, 2 = AUS, 0 = UK and AUS).
- **inplay** : If the market is going in-play (Y = yes, N = no, B = any).
- **years** : The year of the sporting event.
- **exchangeName** : The name of the exchange.

```
Getting historical football events
test.events; //initially empty
*** [ ]
/* requesting all the Italian football events except the ones with Inter, and
retrieving only the Match Odds markets that were going in-play*/
test.getFootballEvents([],["Inter"],["Match Odds"],[],["ITA"],[],[],0,"Y",[],[]);
*** Getting Football Markets...
*** 2 Football Matches found
*** Constructing Football objects...
*** 2 Football objects successfully constructed
test.events;
*** [FootballEvent_0xfffffffb53e6488, FootballEvent_0xfffffffb5449348]
```

- **getEvent** : Returns the requested event from the list of the sporting events.

  <u>Input Parameters</u> :

  - **name** : The name of the sporting event.

Output : A *SportingEvent* object.

- **addMoney** : Deposits money to the main account.

  Input Parameters :

  - **amount** : The amount of money to deposit.

- **stats** : Prints the statistics of the strategy.

- **printEvents** : Prints the sporting events.

- **getResults** : Prints analytical statistics from a historical simulation.

Betting exchange systems have multiple betting markets associated with a particular sporting event. Markets are composed of selections, which customers may back or lay. In that sense, we have:

SportingEvent

The `SportingEvent` represents a sporting event, being a football, tennis, horse race, financial or basketball event. Example: "Man Utd v Arsenal", "Federer v Nadal" and "Kemp 14:10:00 2m4f Hcap". The `SportingEvent` class is simply the base class for all the sports supported in SPORTSBET.

Main attributes

- **id** : The ID of the event.

- **markets** : A list of all the available markets offered for this event.

- **descr** : A description of the sporting event. e.g. "UEFA Champions League ∼ Fixtures 13 March ∼ Malaga v Porto".

- **country** : The country of the event in ISO3 code. e.g. "ITA" for Italy.

- **inplay** : A boolean value. True if the event offers markets going in-play, false otherwise.

- **time** : Holds the current time or the simulation time.
- **startTime** : The expected start time of the event.
- **endTime** : The time the event is finished.
- **backRaised** : An event construct. It can catch and emit any increase at the back prices of any selection in any given market.
- **backFall** : An event construct. It can catch and emit any decrease at the back prices of any selection in any given market.
- **layRaised** : An event construct. It can catch and emit any increase at the lay prices of any selection in any given market.
- **layFall** : An event construct. It can catch and emit any decrease at the lay prices of any selection in any given market.

FootballEvent attributes

- **homeTeam** : The name of the home team.
- **awayTeam** : The name of the away team.
- **homeGoals** : The number of goals the home team has scored.
- **awayGoals** : The number of goals the away team has scored.
- **goal** : An event, triggered whenever a team scores a goal. Holding the name of the team which scored.
- **halfTime** : An event, triggered at the half time of the match.

***TennisEvent* attributes**

- **playerA** : The name of the first player.
- **playerB** : The name of the second player.
- **firstSetEnd** : An event, triggered when the first set ends. Holding the name of the set winner.
- **secondSetEnd** : An event, triggered when the second set ends. Holding the name of the set winner.
- **thirdSetEnd** : An event, triggered when the third set ends. Holding the name of the set winner.
- **fourthSetEnd** : An event, triggered when the fourth set ends. Holding the name of the set winner.
- **fifthSetEnd** : An event, triggered when the fifth set ends. Holding the name of the set winner.
- **currentSetScore** : The current set score of the match.
- **firstSetWinner** : The name of the first set winner.
- **secondSetWinner** : The name of the second set winner.
- **thirdSetWinner** : The name of the third set winner.
- **fourthSetWinner** : The name of the fourth set winner.
- **fifthSetWinner** : The name of the fifth set winner.
- **currentSet** : The number of the current set.
- **playerASets** : The number of winning sets of player A.
- **playerBSets** : The number of winning sets of player B.

***BasketballEvent* attributes**

- **homeTeam** : The name of the home team.
- **awayTeam** : The name of the away team.
- **halfTime** : An event, triggered at the half time of the match.

***HorseRacingEvent* attributes**

- **runners** : A list of the horses running.

***SportingEvent* main functions**

- **placeBet** : A function to place bets with virtual money.

  Input Parameters :

  - **bets** : An array of the bet(s) to be placed.
  - **delay**. A delay to place the bet(s). This is optional.

  Output : An array of bets.

- **placeRealBet** : Same as *placeBet* but with real money.
- **clock** : Returns the current time or the simulation time. The engine can identify a historical associated object from a real time associated object and like that can return the appropriate time.
- **market** : Returns the requested market from the list of the markets.

  Input Parameters :

  - **name** : The name of the market.

  Output : A `Market` object.

- **cancelBet** : Cancel the given bet at the given market.

  Input Parameters :

  - **name** : The market's name.
  - **ID** : The ID of the bet.

- **cancelAllBets** : A function to cancel all the pending bets in every market.

- **controlOdds** : A function to detect an increase or decrease, more than the input percentage, in any selection of the input markets.

  Input Parameters :

  - **markets**: An array with the markets names.
  - **percentage**: The percentage of increase or decrease.

  triggers : *backRaised*, *backFall*, *layRaised*, *layFall*.

  eventValues : `Market` object, `Selection` object, New price, Percentage difference from old price.

- **controlSelectionOdds** : same as *controlOdds* but for specific selections within a market.

  Input Parameters :

  - **market**: The market name.
  - **selections**: An array with the selections names.
  - **percentage**: The percentage of increase or decrease.

  triggers : *backRaised*, *backFall*, *layRaised*, *layFall*.

  eventValues : `Market` object, `Selection` object, New price, Percentage difference from old price.

- **getInplayTime** : Returns how long a sporting event is in-play.

Market

A *Market* represents a market of a sporting event. Example: "Match Odds", "Correct Score", "Set Betting".

Main attributes

- **id** : The ID of the market.
- **name** : The name of the market.
- **selections** : A list of all the selections in this market.
- **goingInplay** : A boolean value. True if the market is going in-play, false otherwise.
- **startTime** : The expected start time of the market.
- **offTime** : The time the market went in-play.
- **endTime** : The time the market was settled.
- **status** : The current status of the market ("ACTIVE", "SUSPENDED", "CLOSED").
- **totalMatched** : The total amount of money that have been matched in this market.
- **inplay** : A boolean value. True if the market is in-play, false otherwise.
- **backOverround** : The overround of the back prices in this market.
- **layOverround** : The overround of the lay prices in this market.
- **liability** : The total liability in this market.
- **profit** : The total net profit/loss in this market.
- **profitAfterCommission** : The total net profit/loss after commission in this market.
- **backFav** : A sorted array with the favourites, according to their back prices in this market.

- **layFav** : A sorted array with the favourites, according to their lay prices in this market.
- **favourite** : The name of the selection with the lowest back price in this market. In case two or more selections have the same lowest back price, the field is null.
- **underdog** : The name of the selection with the highest back price in this market. In case two or more selections have the same highest back price, the field is null.
- **numberOfWinners** : The number of winners in this market.
- **numberOfRunners** : The number of selections in this market.
- **marketUpdated** : An event, triggered whenever the market is being updated.
- **marketInplay** : An event, triggered when the market turns in-play.
- **keepBets** : A boolean value. If true all the pending bets in this market will not be cancelled when the market is suspended.
- **winner** : An array, holding the winner(s) of the market.
- **delay** : The number of seconds delay between submission and a bet actually getting placed when the market is in-play.
- **suspended** : An event, triggered whenever the market is suspended.
- **closed** : An event, triggered when the market is settled.
- **maxLiability** : The maximum liability that occurred over the life of a market.
- **matched** : An array holding the matched bets of the market.
- **unmatched** : An array holding the unmatched bets of the market.
- **cancelled** : An array holding the cancelled bets of the market.

Main functions

- **selection** : Returns the requested selection from the list of the selections.

  Input Parameters :

  - **name** : The name of the selection.

  Output : A *Selection* object.

- **getSelectionIfWinIfLose** : Returns the expected financial outcome if the input selection wins and expected financial outcome if the input selection loses.

  Input Parameters :

  - **id** : the ID of the input selection.

  Output: The profit/loss after commission if the selection wins and the profit/loss after commission if the selection loses.

- **cancelBet** : Cancel the given bet.

  Input Parameters :

  - **ID** : The ID of the bet.

- **cancelAllBets** : A function to cancel all the pending bets of the market.

Selection

A *Selection* represents a possible outcome of a market. Example: "Roger Federer", "2 - 2", "Nadal 2 - 0".

***Selection* attributes**

- **id** : The ID of the selection.
- **name** : The name of the selection.
- **bp** : A list of all the available backing prices in this selection.

- **bv** : A list of all the available backing volumes in this selection.
- **lp** : A list of all the available laying prices in this selection.
- **lv** : A list of all the available laying volumes in this selection.
- **totalMatched** : The total amount matched on this selection (regardless of price).
- **lastMatched** : The last price at which the selection was matched.
- **winPercent** : The winning percentage of the selection.
- **losPercent** : The losing percentage of the selection.
- **updated** : An event, triggered whenever the prices or volumes of the selection have been updated.
- **backUpdated** : An event, triggered whenever the backing prices or volumes of the selection have been updated.
- **layUpdated** : An event, triggered whenever the laying prices or volumes of the selection have been updated.
- **totalBackVolume** : The total volume of the backing prices in this selection.
- **totalLayVolume** : The total volume of the laying prices in this selection.
- **averageBp1** : The average of the best offered backing prices in this selection.
- **averageLp1** : The average of the best offered laying prices in this selection.
- **wabp** : The weighted average of the offered backing prices in this selection.
- **walp** : The weighted average of the offered laying prices in this selection.
- **maxBp1** : The maximum of the best offered backing prices in this selection.
- **maxLp1** : The maximum of the best offered laying prices in this selection.
- **lowLp1** : The lowest of the best offered backing prices in this selection.

- **lowLp1** : The lowest of the best offered laying prices in this selection.
- **ifWin** : The result if the selection wins.
- **ifLose** : The result if the selection loses.
- **Horse Racing Live Selection special attributes**
  - **jockeyName** : The name of the jockey.
  - **ownerName** : The name of the horse's owner.
  - **sex** : The sex of the horse.
  - **officialRating** : The official rating.
  - **daysSinceLastRun** : The number of days since the horse's last run.
  - **age** : The age of the horse.
  - **weight** : The weight of the horse.
  - **trainer** : The name of the horse's trainer.
  - **stallDraw** : The stall number the horse is starting from.
  - **saddleCloth** : The number on the saddle.

***Selection* functions**

- **wom** : The weight of money indicator. It provides a measure of how much money is available to back in comparison to how much is available to lay. In theory, if there is a surplus of back money on a selection at a given odds, the odds at which trades on that selection are matched are likely to fall (e.g. when the winner of a tennis match or horse race is known, the odds of matched trades on that selection fall to 1.01 and then disappear). Likewise, when there is a surplus of lay money, the odds at which trades on that selection are matched are likely to rise (e.g. when the winner of a tennis match or horse race is known, the odds of matched trades on the losing selections rises to 1000). Generally, the way to interpret this value, which is expressed as a percentage, is as follows:

- 0% to 33% means the price is likely to move down.
- 33% to 66% means the direction of price movement is uncertain.
- 66% to 100% means the price is likely to move up.

Bet

The bet is responsible for creating, placing and cancelling a bet. It can either be a Back bet or a Lay bet.

Main attributes

- **profit** : The expected profit of the bet.
- **id** : The id of the bet.
- **liability** : The liability of the bet.
- **odds** : The odds of the bet.
- **volume** : The volume of the bet.
- **volumeMatched** : The volume that has been matched in this bet.
- **matched** : A boolean value. True if the bet is matched, false otherwise.
- **placed** : An boolean value. True if the bet is placed, false otherwise.
- **timePlaced** : The time the bet was placed.
- **timeMatched** : The time the bet was matched.
- **partialMatch** : A boolean value. True is the bet is partial matched, false otherwise.
- **selection** : The selection on which the bet refers.
- **market** : The market on which the bet refers.
- **marketName** : The name of the market the bet refers.

- **selectionName** : The name of the selection the bet refers.
- **unmatched** : A boolean value. True if the bet is unmatched, false otherwise.
- **betMatched** : An event, triggered when the bet is matched.
- **betPlaced** : An event, triggered when the bet is placed.
- **cancelled** : An event, triggered when the bet is cancelled.

Time

The `Time` handles the simulation and current time of a sporting event.

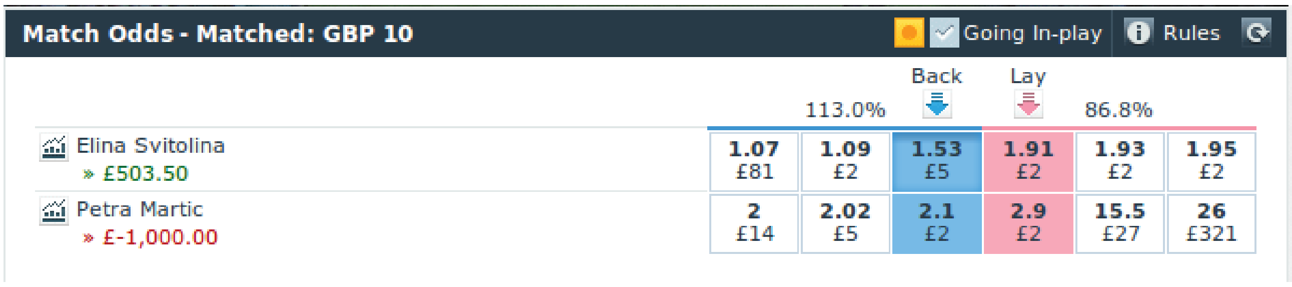
3.4.3 UBEL's Global objects

- **currentClock** : The current time.
- **simulationClock** : The simulation time.
- **betfair** : The `betfair` object holds specific requirements of the Betfair exchange, such as acceptable range of prices, commission and bet types (see Appendix B). Furthermore, provides an advanced interface to Betfair API.

3.4.4 UBEL's Global Functions

Hedging Functions

greenStake : A popular technique used by many traders is to try to "Green-up" after making a successful trade. "Greening-up" is a means of making a profit whatever the outcome of a market is. Suppose we have a tennis match between Elina Svitolina and Petra Martic. Suppose we back Elina Svitolina @ 1.53 (the best backing odds available now) with £1000.



Match Odds - Matched: GBP 10 Going In-play Rules

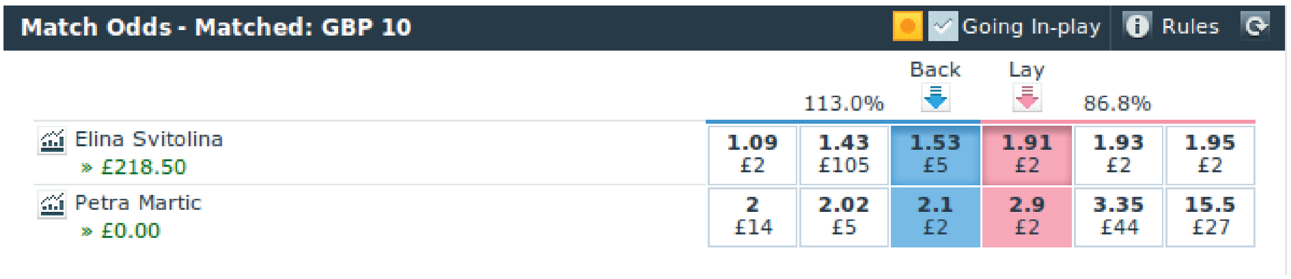
			Back	Lay		
		113.0%			86.8%	
Elina Svitolina » £503.50	1.07 £81	1.09 £2	1.53 £5	1.91 £2	1.93 £2	1.95 £2
Petra Martic » £-1,000.00	2 £14	2.02 £5	2.1 £2	2.9 £2	15.5 £27	26 £321

So under Elina Svitolina we have the sum we would win if we left the bet like that until the end and Svitolina wins the match. On Martic is the sum we will lose if Martic wins the match. Assume that Svitolina will break Martic fast. The odds will drop to around 1.3 - 1.4 for Svitolina. By then we have two options :

- Sell the bet to make profit regardless of the outcome (let the profit on Svitolina or spread the profit on the two tennis players)
- Risk and leave the bet until the end of the set when if Svitolina takes it the odds should be around 1.2 for her and we would have more profit secured but more pressure on our shoulders.

At the 1st option there are two possible outcomes:

1. Lay Svitolina @ 1.3 for £1000.

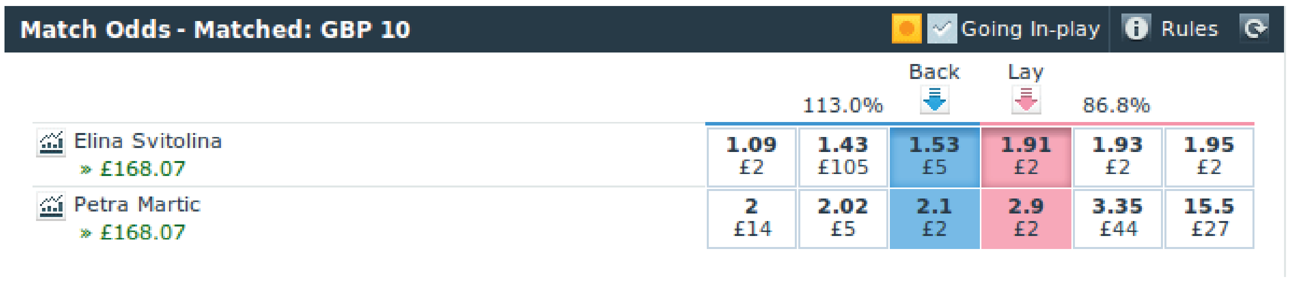


Match Odds - Matched: GBP 10 Going In-play Rules

			Back	Lay		
		113.0%			86.8%	
Elina Svitolina » £218.50	1.09 £2	1.43 £105	1.53 £5	1.91 £2	1.93 £2	1.95 £2
Petra Martic » £0.00	2 £14	2.02 £5	2.1 £2	2.9 £2	3.35 £44	15.5 £27

That means that we win £218.5 if Svitolina wins the match or we will lose nothing if Martic wins.

2. Second option is to spread the profit equally on both players. In our case we will lay Svitolina @ 1.3 for £1176.92.



Match Odds - Matched: GBP 10 Going In-play Rules

		113.0%	Back	Lay	86.8%	
Elina Svitolina » £168.07	1.09 £2	1.43 £105	1.53 £5	1.91 £2	1.93 £2	1.95 £2
Petra Martic » £168.07	2 £14	2.02 £5	2.1 £2	2.9 £2	3.35 £44	15.5 £27

That means that we win £168.07 regardless of the outcome of the match.

So the function `greenStake` is calculating the stake to spread the profit across all the selections. This function works assuming you have only one bet matched for a selection.

Input Parameters :

- **matchedBetPrice** : The price of the matched bet.
- **matchedBetStake** : The stake of the matched bet.
- **newPrice** : The new price of the selection.
- **exchangeName** : The name of the exchange.

Odds Control functions

getNextBackOdds : Returns the next acceptable back price which is after the given one.

Input Parameters :

- **price** : The input price.
- **exchangeName** : The name of the exchange.

ticks : Returns the next acceptable price which is X ticks away from the given one.

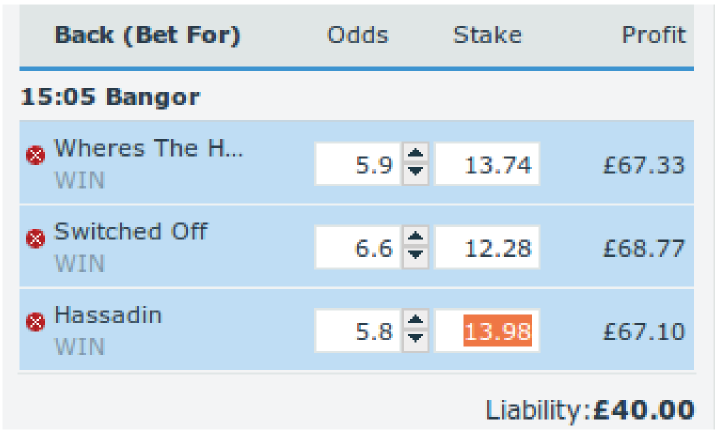
Input Parameters :

- **price** : The input price.
- **ticks** : The number of ticks.
- **exchangeName** : The name of the exchange.

Dutching functions

Dutch betting is a very powerful way of increasing the probability of a win scenario. This involves spreading your stake over more than one selection. The amount bet on each selection is calculated so that the same amount of profit is achieved in case any of the backed selections it is a winner. In that sense we have:

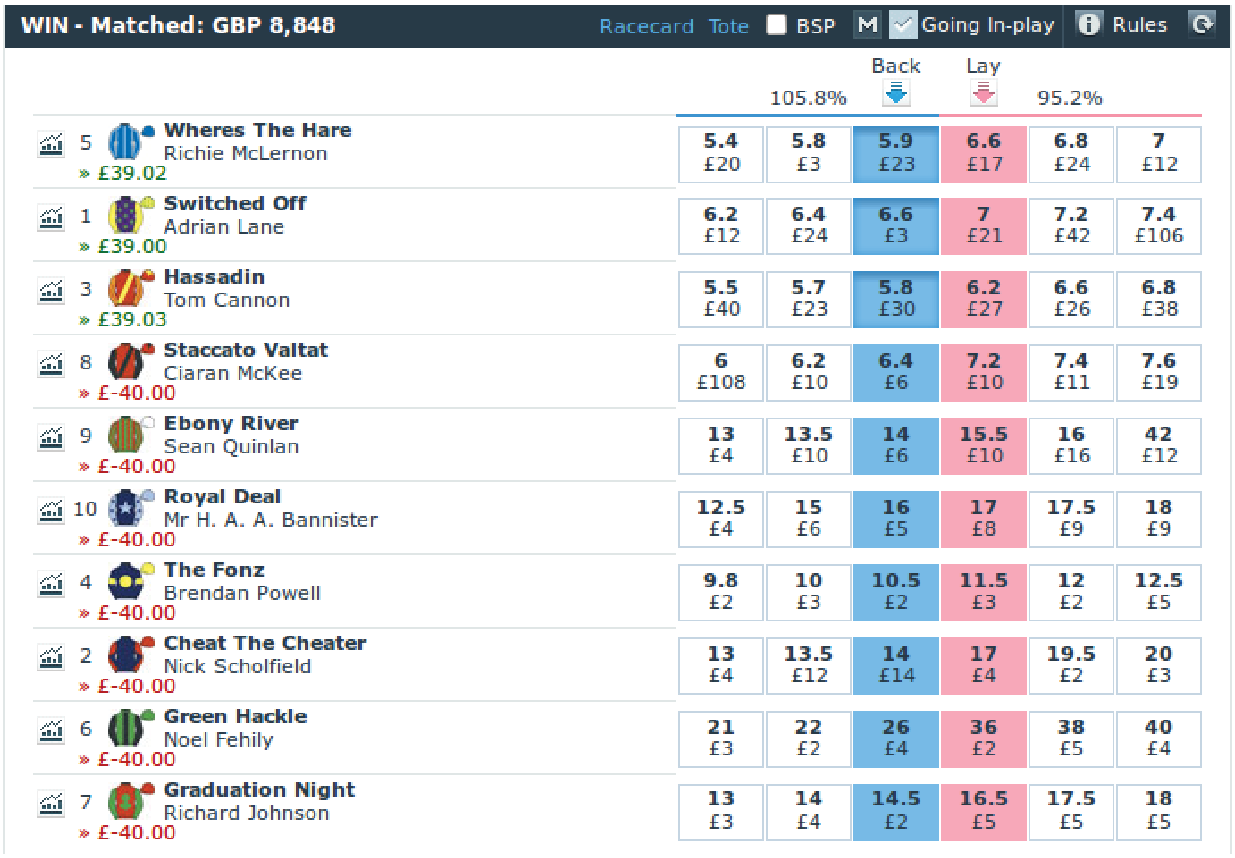
dutch : Backing the given selections of the given market and sporting event for the given stake. If the stake needed to back all the given selections is more than the input stake, no bet is placed. Suppose we have the horse race "Bangor 15:05:00 3m Hcap Hrd" and we want to dutch the horses "Hassadin", "Switched Off" and "Wheres The Hare" for £40. Then the `dutch` function will place three back bets:



Back (Bet For)	Odds	Stake	Profit
15:05 Bangor			
Wheres The H... WIN	5.9	13.74	£67.33
Switched Off WIN	6.6	12.28	£68.77
Hassadin WIN	5.8	13.98	£67.10

Liability:**£40.00**

And the result would be:



WIN - Matched: GBP 8,848 Racecard Tote BSP M Going In-play Rules

				105.8%	Back	Lay	95.2%	
5	Wheres The Hare Richie McLernon	» £39.02	5.4 £20	5.8 £3	5.9 £23	6.6 £17	6.8 £24	7 £12
1	Switched Off Adrian Lane	» £39.00	6.2 £12	6.4 £24	6.6 £3	7 £21	7.2 £42	7.4 £106
3	Hassadin Tom Cannon	» £39.03	5.5 £40	5.7 £23	5.8 £30	6.2 £27	6.6 £26	6.8 £38
8	Staccato Valtat Ciaran McKee	» £-40.00	6 £108	6.2 £10	6.4 £6	7.2 £10	7.4 £11	7.6 £19
9	Ebony River Sean Quinlan	» £-40.00	13 £4	13.5 £10	14 £6	15.5 £10	16 £16	42 £12
10	Royal Deal Mr H. A. A. Bannister	» £-40.00	12.5 £4	15 £6	16 £5	17 £8	17.5 £9	18 £9
4	The Fonz Brendan Powell	» £-40.00	9.8 £2	10 £3	10.5 £2	11.5 £3	12 £2	12.5 £5
2	Cheat The Cheater Nick Scholfield	» £-40.00	13 £4	13.5 £12	14 £14	17 £4	19.5 £2	20 £3
6	Green Hackle Noel Fehily	» £-40.00	21 £3	22 £2	26 £4	36 £2	38 £5	40 £4
7	Graduation Night Richard Johnson	» £-40.00	13 £3	14 £4	14.5 £2	16.5 £5	17.5 £5	18 £5

Input Parameters :

- **sportingEvent** : The sporting event.
- **market** : The market name.
- **selections** : The selections names.
- **stake** : The stake.
- **exchangeName** : The name of the exchange.

Execution functions

monitor : Start monitoring all the available markets of all the sporting events in a SportingEvent-Collector.

Input Parameters :

- **collectors** : A list of the SportingEventCollector objects.

- **when** : When to start monitoring the sporting events, using as a base time the expected start time of the event.

- **updateRate** : The update rate of the markets.

- **saveData** : A boolean value. If true the market prices and results will be saved in a database.

- **exchangeName** : The name of the exchange from which the prices will be requested.

Requesting and Monitoring Horse Racing markets from Betfair

```
//constucting a SportingEventCollector object

var test = SportingEventCollector.new(7,"Betfair");

test.liveStrategy("myFirstStrategy","test");


/*requesting all the tennis "Match Odds" in-play markets*/

Betfair.getLiveTennisEvents("","","Match Odds","","","","",0,"Y",0,0);


/*requesting to monitor from Betfair all the sporting events of

"test", 2 minutes before the match start and save all the prices

and results.*/

monitor([test], -2min, 1s, true, "Betfair");
```

For more functions of UBEL the reader is referred to Appendix A.

3.5 Strategies examples

10 minutes no Goal

10 minutes no goal

```
function tenMinutesNoGoal(var sportingEvent){

var homeTeam = sportingEvent.market("Match Odds").selection(sportingEvent.homeTeam);

var awayTeam = sportingEvent.market("Match Odds").selection(sportingEvent.awayTeam);

  at(sportingEvent.startTime - 30min == sportingEvent.clock){

    if(abs(homeTeam.bp[0] - awayTeam.bp[0]) <= 1.5){

      var layBet = Lay.new("First Goal", "0 - 10", 10, "Betfair");

      sportingEvent.placeBet([layBet]);

    };

  };

};
```

Trailing Stop Loss

The trailing stop loss strategy can be used to automatically close a trade in order to limit the losses of the strategy or to lock in a profit. After an initial bet is matched, the stop loss function can automatically adjust the price of the opposing bet, to close the position when the market price moves too far against you. The actual trigger price is measured in ticks from the matched price of original bet. The inputs of the strategy are: the sporting event, the market's and selection's name, the desired number of ticks in order to exit the strategy with a loss, the desired number of ticks in order to lock in a profit and the type of bet (back or lay).

Trailing Stop Loss

```
function stopLoss(var sportingEvent, var _market, var _selection, var maxTicks, var minTicks, var betType){
var market = sportingEvent.market(_market);
var selection = market.selection(_selection);
  if(betType ="Back"){
    var backBet = Back.new(_market, _selection, 20, "Betfair");
    var layBet = Lay.new(_market, _selection, ticks(backBet.odds,-minTicks,"Betfair"), 20, "Betfair");
    var bets  = sportingEvent.markets[0].placeBet([backBet,layBet]);
    at(bets[0].matched){
      at(selection.layUpdated?(var lp, var lv)){
        if(!bets[1].matched){
          var checkStop = ticksDiff(lp, bets[0].odds, "Betfair");
          if(checkStop >= maxTicks){
            bets[1].odds = lp;
          }
          else if(checkStop > -minTicks && checkStop < 0){
            bets[1].odds = ticks(bets[1].odds, -(checkStop + minTicks), "Betfair");
          };
        };
      };
    };
  }
  else if(betType = "Lay"){
    var layBet = Lay.new(_market, _selection, 20, "Betfair");
    var backBet = Back.new(_market, _selection, ticks(layBet.odds,maxTicks,"Betfair"), 20, "Betfair");
    var bets  = sportingEvent.markets[0].placeBet([layBet,backBet]);
    at(bets[0].matched){
      at(selection.backUpdated?(var bp, var bv)){
        if(!bets[1].matched){
          var checkStop = ticksDiff(bp, bets[0].odds, "Betfair");
          if(checkStop <= -minTicks){
            bets[1].odds = bp;
          }
          else if(checkStop > 0){
            bets[1].odds = ticks(bets[1].odds, (checkStop + maxTicks), "Betfair");
          };
        };
      };
 };};};
```

Chapter 4

SPORTSBET Architecture

Betting exchange markets are complex systems consisting of entities interacting and evolving in an uncertain environment. Their modelling and simulation requires the use of technology in order to model the various actors participating in a market and their interaction with trading strategies. This chapter presents the SPORTSBET architecture. We provide a checklist of issues which were considered during the design of SPORTSBET architecture and we show how our structured solution meets all of the technical and operational requirements, while optimising common quality attributes. We discuss why the design of a betting exchange market simulator it is a challenging task due to the operational complexity and computationally costly decision support. We present our approach in which the complexity of the betting exchange market functionality is decomposed into relatively simple tasks and processes and finally we present the simulator architecture.

4.1 Design philosophy of SPORTSBET

Like any other complex structure, SPORTSBET was built on a solid foundation. The design philosophy of SPORTSBET was:

- List the desired goals, properties and functional utilities of the system in rank of importance.

- Derive design implications of the desired goals/properties/utilities of the system.
- Search for the appropriate methods/technology for designing appropriate architecture.
- Repeat the above steps until we find a feasible and economical architecture that performs as desired.

4.1.1 List of desired properties/utilities and their implications

The typical objective of a betting system is to identify markets to bet by applying a predetermined set of rules (betting strategy) otherwise, if there are no candidates to which the rules apply, it is of no use. These rules can be based upon previous analysis of form variables and ratings, backtesting etc.

To understand better the desired goals we will discuss first how an advanced betting process looks like (see Figure 4.1) and how sports traders workstations should be.

1. Select a sporting event from the universe.
2. Identify opportunities within the sporting event by applying : Predefined rules, Rating methods, Predictive models and Third party advice and rating.
3. Capture market(s) prices of the selected sporting event.
4. Determine if there are any “value bets”. A “value bet” occurs when the true chance of a selection within a market to win or lose is greater than the one estimated and offered.
5. Betting decision.
6. Execute bet(s).

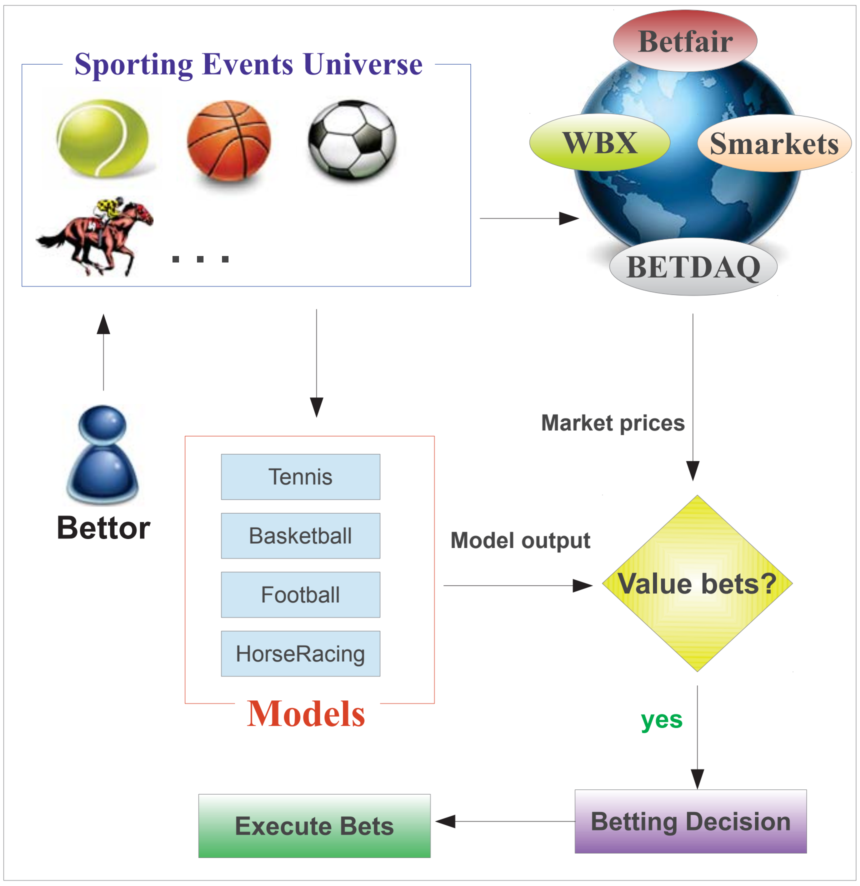


Figure 4.1: Betting process example

A sports traders workstation must offer the following benefits:

1. A simplified but sophisticated workspace that offers real-time account balance and activity monitoring.
2. A comprehensive suite of sophisticated mathematical risk-assessment and price-measuring tools.
3. Must allow the input of several incoming feeds (in-play events, video feeds etc.).
4. An optional virtual account, linked to the production account, which allows the trader experiment in a simulated environment.

The desired properties are :

Functionality : allows the generic specification, back-testing, optimisation and execution of parameterised automated trading strategies for betting exchange markets. It must be able to test the strategies against historical or real time data with either real or virtual capital and support the automatic optimisation of the strategy parameters.

Performance : must perform complex calculations and get answers as quickly as possible. This is very important, as a strategy can depends on complex numerical calculations.

Provability : you must be able to prove the correctness of your algorithms. Mistakes will be very expensive when trading with real money.

Low Latency : needs a very fast communication technology. Low latency is crucial for trading systems.

Prototyping/Development speed : in some situations it is critical to develop and test your strategy as quickly as possible in order to exploit the window of opportunity.

Economical : system must be affordable.

Scalability : should be able to accommodate growing demand on betting exchange platforms, sports, modules and data.

Robustness : must operate when a few components fail or it must restore system to last a working state in case there is a complete fail.

User Friendly : must be a cross-platform and can be very easily used by people with no programming background.

Integration : must be able to interface to many different information sources and back end systems.

Concurrency : must be able to take the maximum advantage of an expensive multi-core machine whether for latency or raw throughput.

The basic implications are :

- need of a complex event processor capable of handling all the events. It must allow the synchronisation of multiple data streams.
- need of stream processing to analyse multiple streams from multiple data sources and trigger events for the corresponding sporting events and their associated markets.
- need of a fairly simple generic betting language capable of defining simple and complex betting strategies for a wide range of sports fairly simple.
- need of a heavily optimised language to perform complex calculations really fast.
- need an efficient optimisation algorithm to automatically refine the strategy parameters. Must be fast, accurate and avoid overfitting.
- the language must be able to support easily any new sport or betting exchange platform.
- need to replicate the exact conditions of the sporting event and each associated markets. In addition to replicate the exact conditions between sporting events as well.
- allowing the parallel simulation of different markets to take the advantage of a multi-core system.
- need of a database system to store the betting exchange data.
- the system must support Time-driven, Request-driven, and Event-driven interactions.

  Time-driven example : Collect all the Match Odds markets of the daily football events every morning at 8 A.M. provided that the best back price of the favourite is more than 2.5.

  Request-driven example : Back now Arsenal at the best available odds for £10 provided that the price is less than 1.5.

  Event-driven example : Whenever a goal is being scored, cancel all my pending bets in this sporting event.

4.1.2 Desired properties solutions

In order to meet all the desired properties mentioned before, SPORTSBET was built on the open-source platform of URBI. Two unique features of URBI made it suitable for our purposes. First, URBI includes UrbiScript that features support for advanced concurrency abstractions and event-based programming using event-calculus-like operators. Second, URBI has its own engine capable of handling several forms of process parallelism. We extended the UrbiScript language with classes to support the definition of betting strategies and we augmented UrbiEngine with processes which model the operation of betting exchange markets. In addition, UBEL includes predefined mathematical functions related to betting such as: "dutching", "greening up" etc.

The SPORTSBET framework supports the parallel simulation and/or live monitoring of many different sporting events and their associated markets. Betting strategies are specified using UBEL as sets of concurrent processes which make use of event-calculus-like operators (e.g. *at, whenever, watch etc.*). These strategies can place or cancel many bets in parallel across different markets. Useful in-play events may be explicitly provided (e.g. through a feed provided by a company such as Opta[1]) or may be inferred from analysing the price data across different markets. For example it is possible to infer winning of a set in tennis from the Set Betting market. Moreover there is an option to visualize the strategy execution over the course of each sporting event.

We note that in addition to live or historical event streams, SPORTSBET can also easily accept synthetic input streams from e.g. hierarchical Markov chain simulations of tennis matches [89].

4.2 SPORTSBET components

The overall architecture of SPORTSBET can be seen in Figure 1.2. SPORTSBET consists of many components and operations, each of which addresses the desired properties and implica-

[1]See http://www.optasports.com

tions mentioned previously. The basic components are:

UBEL

A generic betting strategy specification language. It is responsible for simplifying the definition of both simple and complex betting exchange trading strategies for a wide range of sports.

Complex Event Processor (Urbi Engine)

Responsible for analysing event data (multiple streams of events) either in real-time or historical basis, in order to find and react to events, relevant to the current betting strategy or sporting event, within the event cloud. Events can also be emitted from the strategy execution itself but also from the user in real time. It supports Event-driven, Time-driven and Request-driven interactions.

Evaluation Tool

It is responsible for quantifying the strategy performance. It outputs statistics such as Return on Investment and Maximum Drawdown.

Optimisation platform

It is responsible for the automatic optimisation of the given strategy. The optimisation algorithm is efficiently searching the solution space of the parameterised strategy and finds optimal or near optimal solutions.

User Interface

A simple, intuitive and responsive user interface, ultimately minimising the effort and time taken to perform a desired task. Includes an editor designed specifically for UBEL.

Database

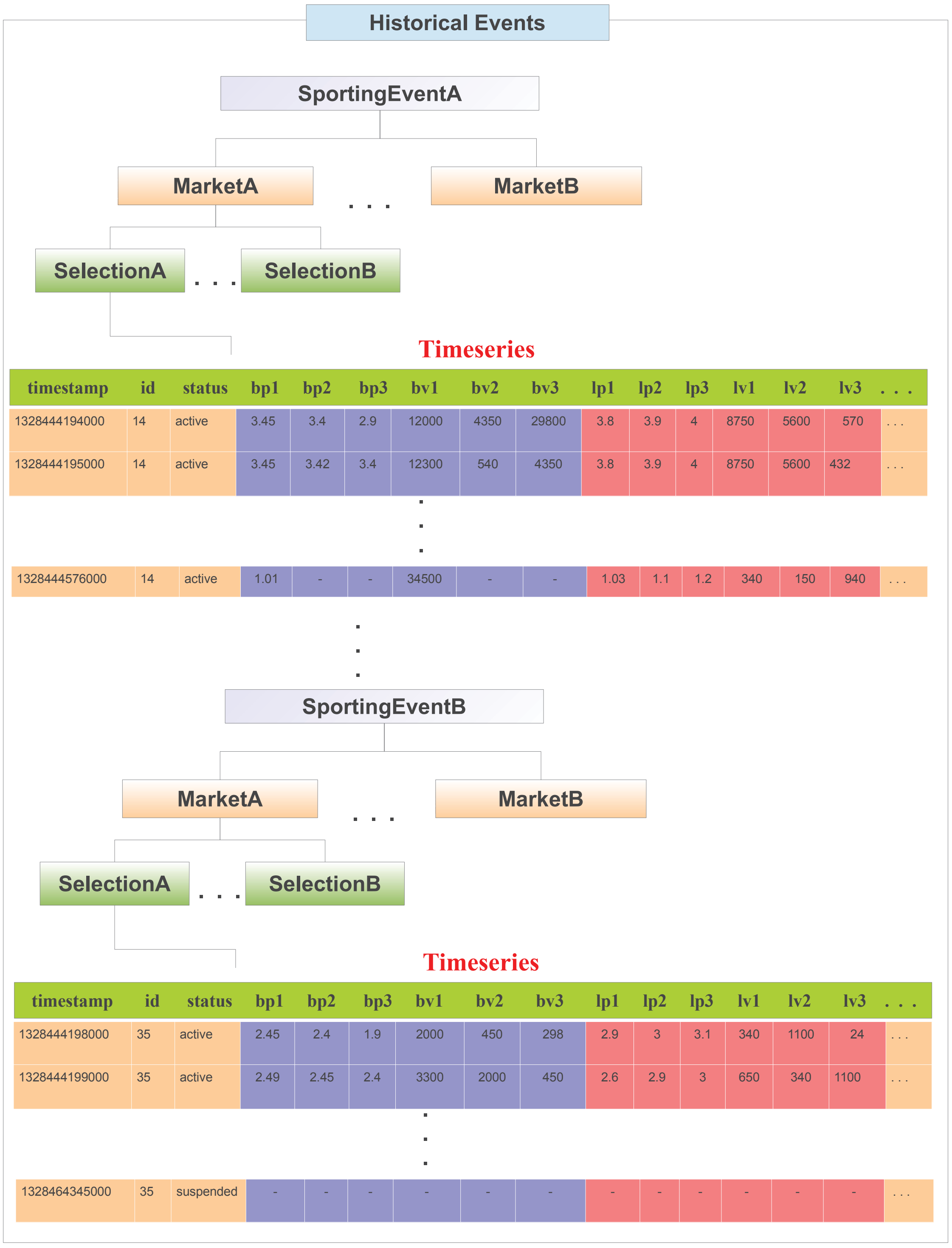
Responsible for storing efficiently very large numbers of sporting events records and provide a very quick and easy way to extract the required information whenever is needed.

4.3 Betting exchange market simulator

4.3.1 Complexity

Several vendors exists to meet the challenge of back-testing and simulation in finance so traders can test their trading strategies[2]. However, nothing similar exists for betting exchange markets. The betting exchange market simulator use historical data (time series) to dynamic reconstruct the market and replicate the exact conditions in time. The Figure 4.2 shows the structure of a collection of historical sporting events. Each selection within a market has a corresponding historical time series dataset. The time series can be updated per millisecond, second, minutes etc. depends on what the user has specified. So each selection can have thousands lines of time series data. During the dynamic market reconstruction and simulation of a market all the time series for the selections in a market must be synchronised. In case there is a need to simulate more than one market within a sporting event then all the selections from all the markets must be synchronised. Finally, when there are several sporting events to be simulated all the markets from all sporting events must be synchronised when necessary.

[2]http://www.amibroker.com/, http://www.metastock.com/, http://www.tradestation.com/



timestamp	id	status	bp1	bp2	bp3	bv1	bv2	bv3	lp1	lp2	lp3	lv1	lv2	lv3	. . .
1328444194000	14	active	3.45	3.4	2.9	12000	4350	29800	3.8	3.9	4	8750	5600	570	. . .
1328444195000	14	active	3.45	3.42	3.4	12300	540	4350	3.8	3.9	4	8750	5600	432	. . .
						⋮									
1328444576000	14	active	1.01	-	-	34500	-	-	1.03	1.1	1.2	340	150	940	. . .

⋮

timestamp	id	status	bp1	bp2	bp3	bv1	bv2	bv3	lp1	lp2	lp3	lv1	lv2	lv3	. . .
1328444198000	35	active	2.45	2.4	1.9	2000	450	298	2.9	3	3.1	340	1100	24	. . .
1328444199000	35	active	2.49	2.45	2.4	3300	2000	450	2.6	2.9	3	650	340	1100	. . .
						⋮									
1328464345000	35	suspended	-	-	-	-	-	-	-	-	-	-	-	-	. . .

Figure 4.2: Structure of historical sporting events collection

Figure 4.3 shows the complexity of the multiple time series data synchronisation. A sporting event can have multiple markets and a market can have more than 20 selections (e.g. horse racing markets) to be synchronised. So in case a trader wants to simulate thousands of markets and replicate the exact conditions in time the synchronisation complexity can be really high.

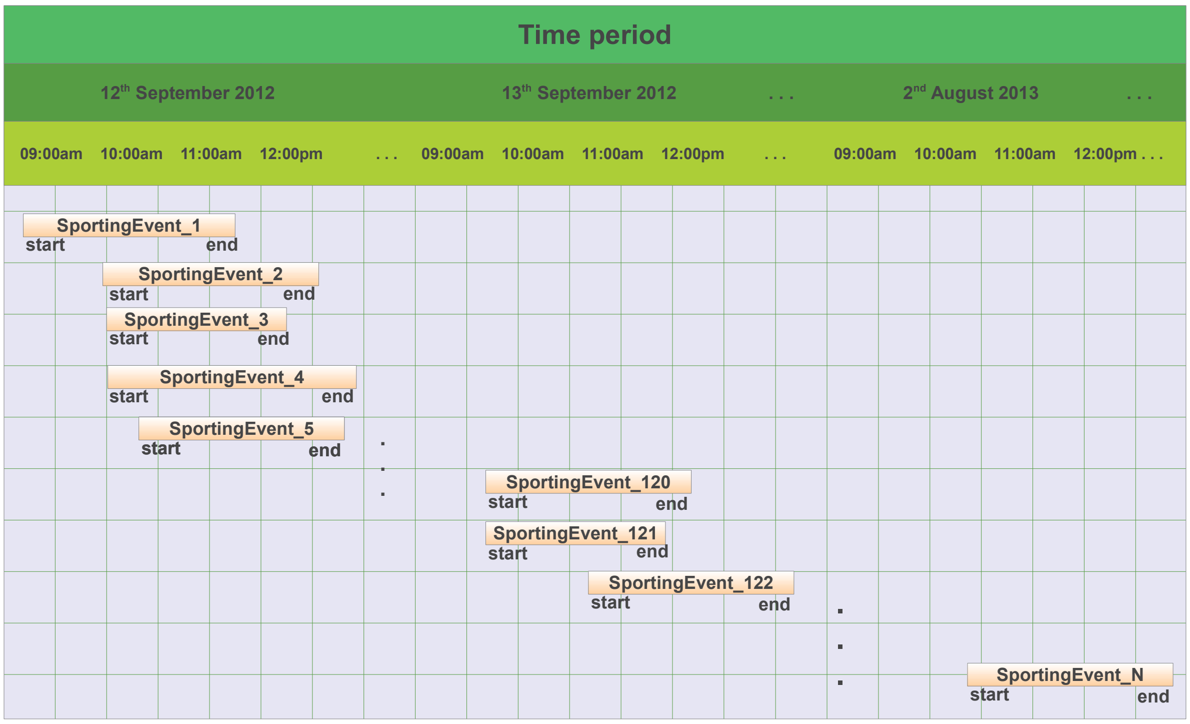


Figure 4.3: Synchronisation complexity

4.3.2 SPORTSBET synchronisation approach

Figure 4.4 shows how SPORTSBET can synchronise multiple data streams. Each sporting event can have many event handlers associated with it in order to catch events such as goal being scored, set finished etc. For each market within a sporting event there is an event handler responsible to catch event streams for this market and update it. In addition, for each selection within a market there is a thread responsible to update the selection's status, such as the available lay and back prices for this selection. All the selections in a market and all the markets within a sporting event are updated concurrently using the powerful mechanisms of URBI's Kernel.

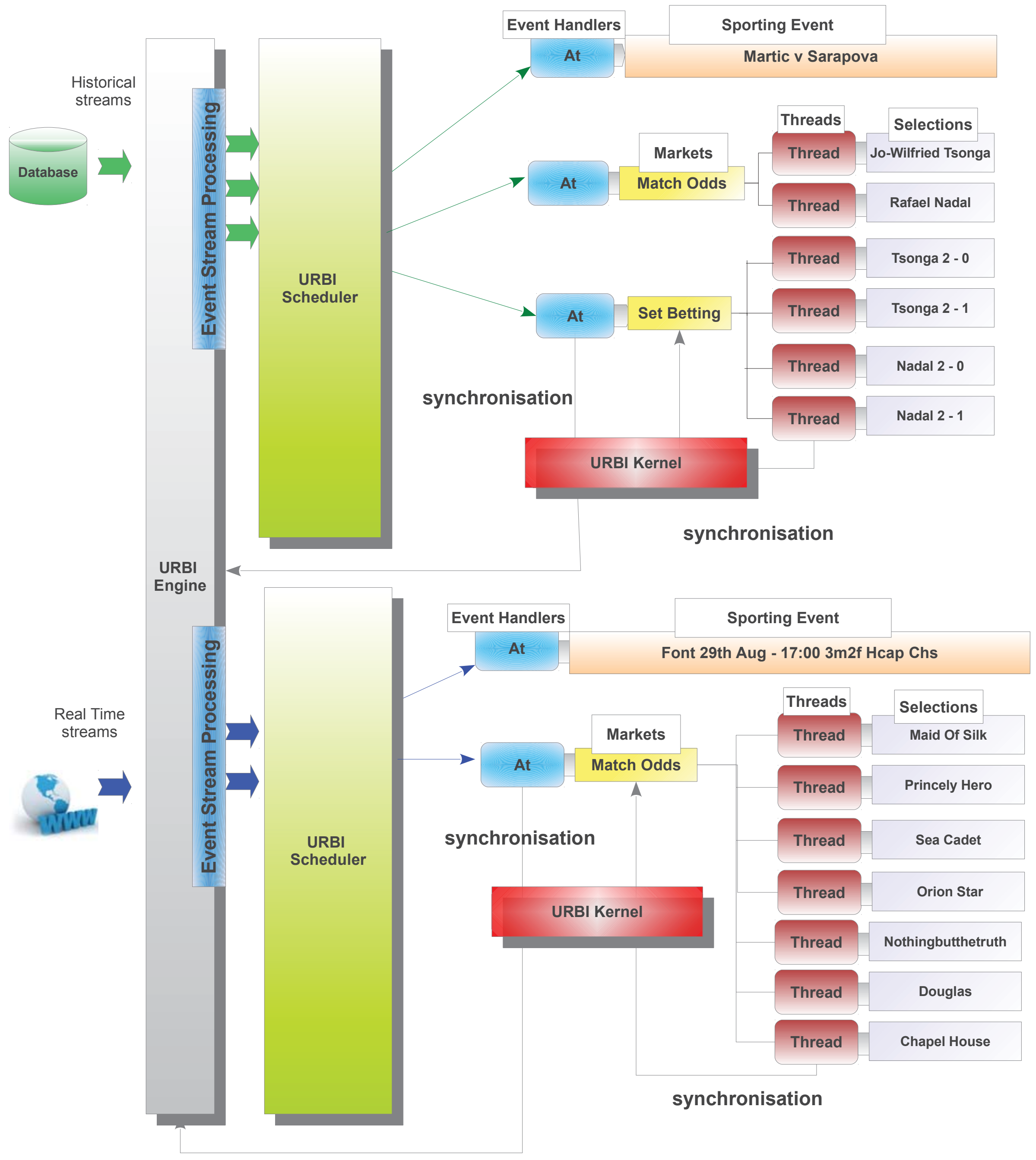


Figure 4.4: SPORTSBET synchronisation of multiple data streams

4.3.3 Back-testing and Simulation challenges

The back-testing process requires to replicate the market conditions of the time in question in order to get an accurate result. That means that all the markets must be executed at the exact time and order they were scheduled in real time. Failing to do that the evaluation results will not be accurate. This is because although each sporting event is independent from each other, when executing a trading strategy the result from one market can affect the trading strategy in forthcoming markets or even some of the evaluation measures. Let us assume we have 4 sporting events and that each one has only one market. We have a trading strategy and we want to test it against those markets. In Figure 4.5 we can see the exact time occurrence of each sporting event and the result and liability of the strategy in each one of them individually.

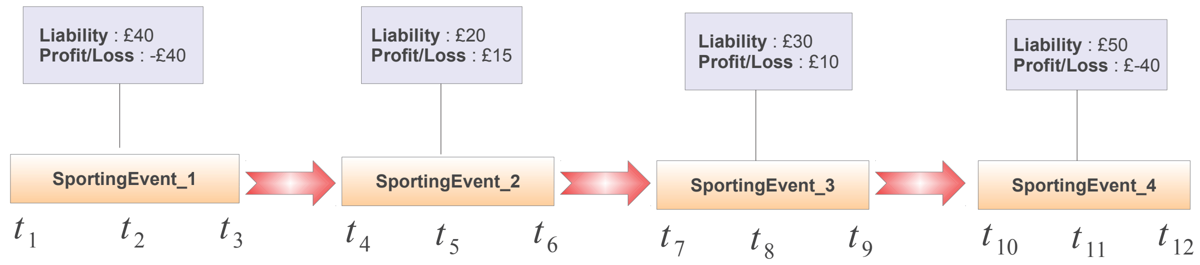


Figure 4.5: Back-testing scenario

Figure 4.6 shows the back-testing process and results if we simulate the sporting events respecting their timestamps. Assuming that our initial account balance is £100 as we can see we lost £55 and the maximum risk we took when executed this strategy was £60.

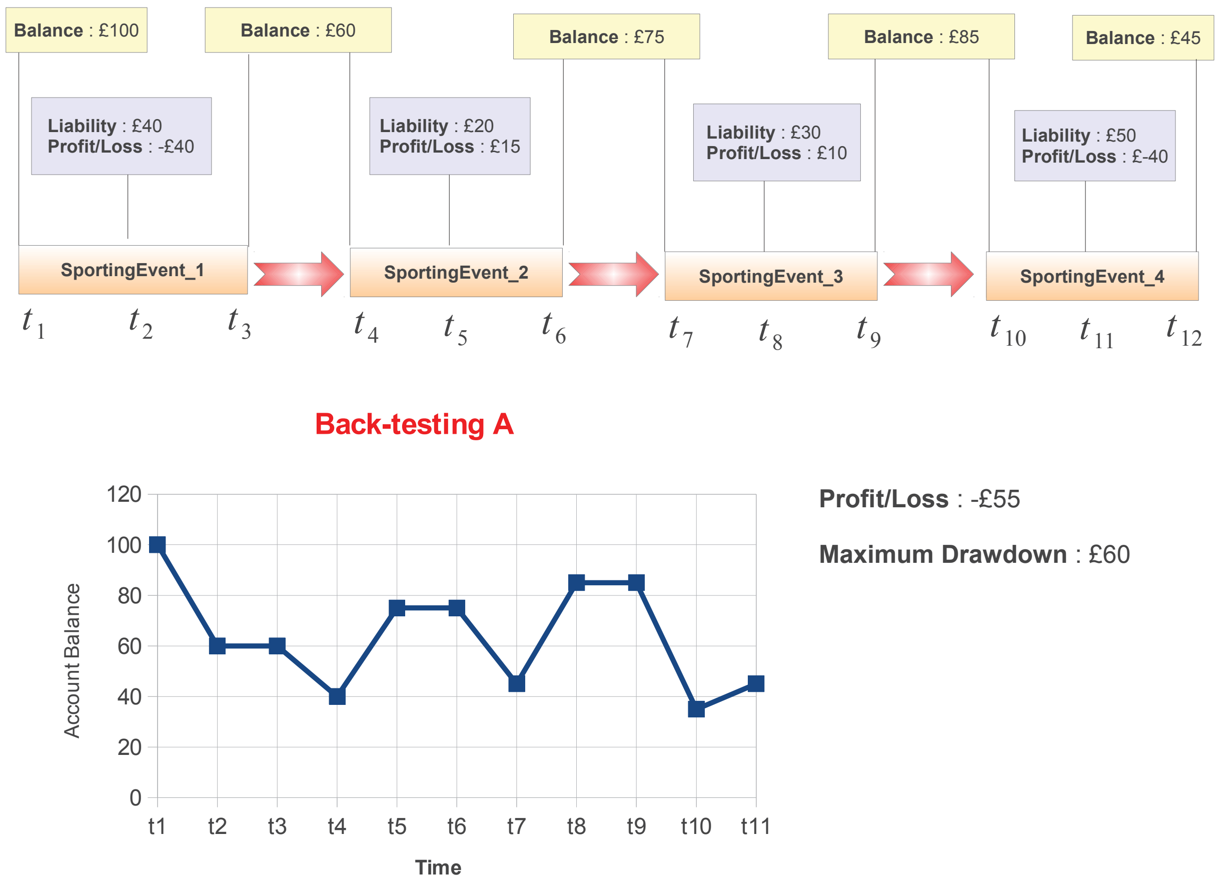


Figure 4.6: Back-testing process and results (respecting the sporting events timestamps)

In case during the back-testing process the order of each sporting event started and finished is not respected, someone would expect the profit/loss of the strategy to be the same and indeed that is the case. But how about the maximum risk we took? Figure 4.7 shows the back-testing process and results when the sporting event timestamps are not respected.

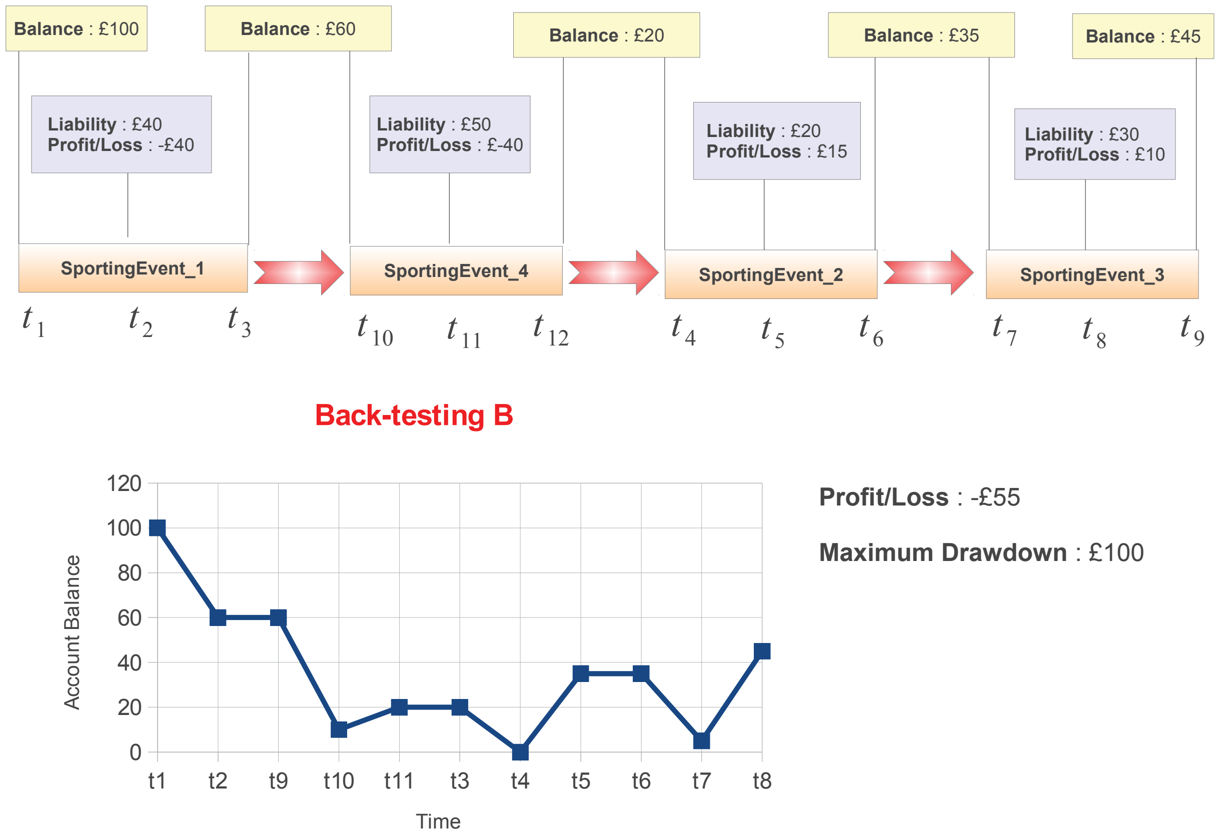


Figure 4.7: Back-testing process and results (not respecting the sporting events timestamps)

As we can see the maximum risk we took is higher comparing with the risk we took in first case. So results like that can be misleading. Moreover, in some cases the trader might want to have as stop criteria of the strategy execution the value of the maximum drawdown (MDD). So the trader might want to stop the strategy execution if the $MDD > £60$, all these things must be taken into account when preparing the back-testing environment.

In order to overcome that, the markets are sorted by timestamp and then by using some powerful UrbiScript constructs and a global simulation clock we are able to replicate the exact conditions in time. However, a back-testing process like that can be very time consuming when we have thousands of markets to simulate. To deal with this issue SPORTSBET allows the parallel simulation of all the markets. To do that we have used the concurrency mechanisms of URBI and for every sporting event we have assigned a local simulation clock. When all the markets have been simulated, we have implemented a process which is passing through the attributes of each constructed market object and extracting the same results with the sorting approach. However, when a trader is using the parallel back-testing process (s)he can not use measures such as MDD during the strategy execution.

4.3.4 Betting challenges

The betting process requires to capture market(s) prices of the selected sporting event and continuously check if there is a match between the bet's price and the selection's price for the specific market. That means when a bet is being placed it must stay in the "background" and be always ready to react to events. These events can be external (market prices changed) or internal (user increased the volume of the bet, canceled the bet etc.). Figure 4.8 shows the possible states of a bet and their transitions.

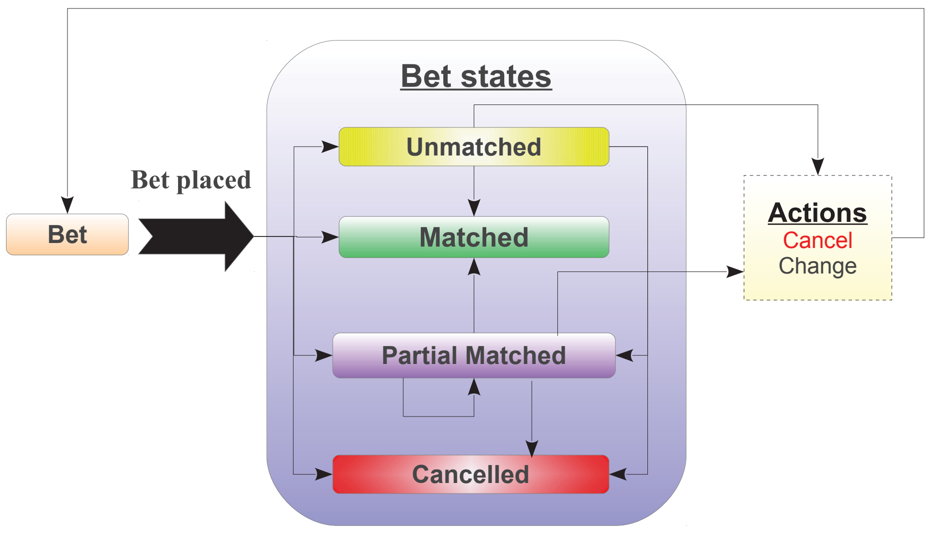


Figure 4.8: Possible states and transitions of a Bet

After a bet has been placed it can be in any of the following 4 states:

- **Matched** : The bet is matched and no other transition is allowed.
- **Unmatched** : When a market is active and a bet is unmatched then the bet is possible to make a transition to any other state.

- **Partially matched** : When a bet is partially matched the remaining unmatched bet can make a transition to any other state. The matched part of the bet cannot be modified.
- **Canceled** : When a bet is canceled the bet does not exist any more.

Figure 4.9 shows how SPORTSBET can constantly check when a bet can be matched or changed its state.

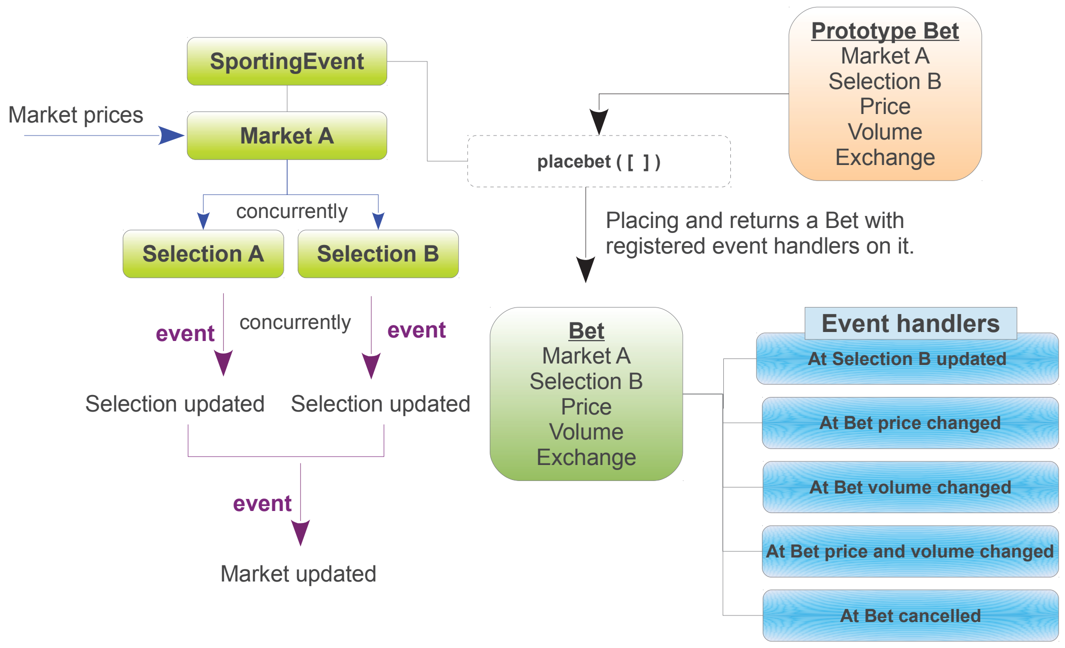


Figure 4.9: SPORTSBET betting solution

The user is creating first a Bet object (Prototype Bet) by choosing the market name, the selection name, the volume, the exchange and the price (in case the price is not given, the engine will choose the best available price for the given selection at the moment the bet will be placed). The next step is to place this bet by using the method `placeBet` of the SportingEvent and passing as input an array of bets (in this case the array will have only one bet). When this method is executed it will automatically create a clone of the prototype bet and associate it with the given Market and the Selection of the SportingEvent. In addition, the cloned bet will have registered event handlers to catch any internal or external event and react as properly. As an example if the price of a lay bet is changed by the user then the event handler which is responsible to catch price changes of the bet will be activated and then it will first update the state of the bet(expected profit, liability, etc.) and then pass all the new information to the

Market. The Market will use this new information to update as an example the total liability of the user in this Market. The prototype bet can be placed as many times the user wants.

The problem of getting matched with already matched volumes

When a trader is testing a strategy against historical or live data using virtual capital there are some cases where a bet can be matched with already matched volumes. Figure 4.10 shows this problem.

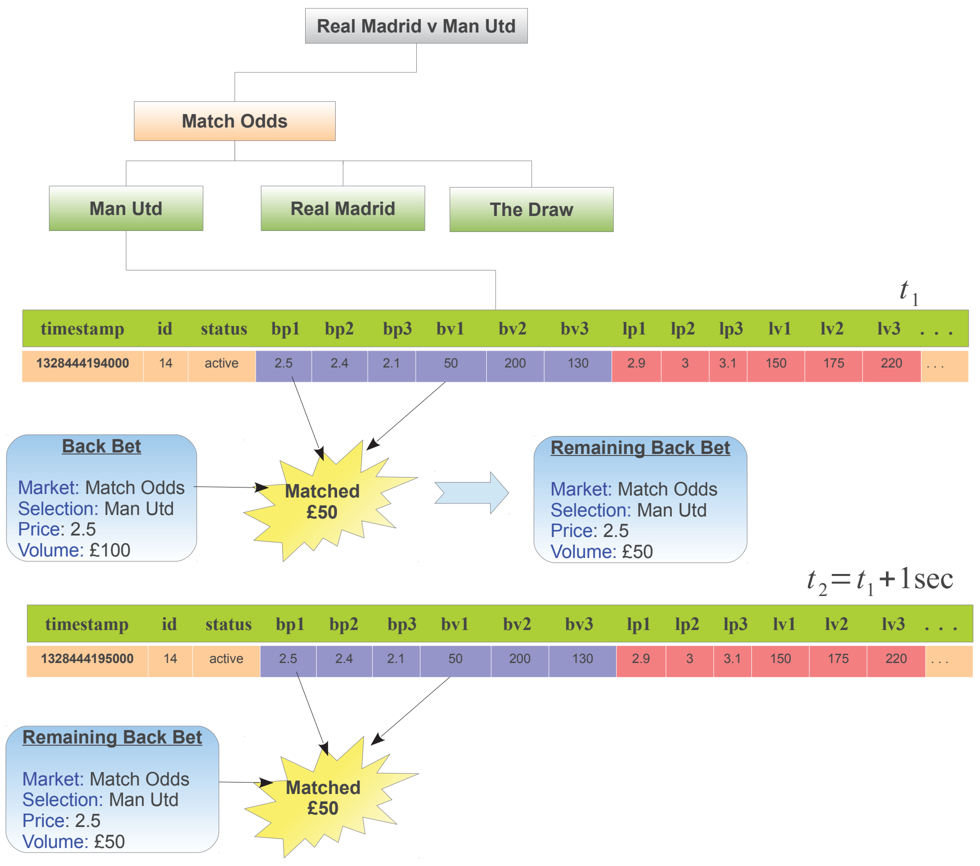
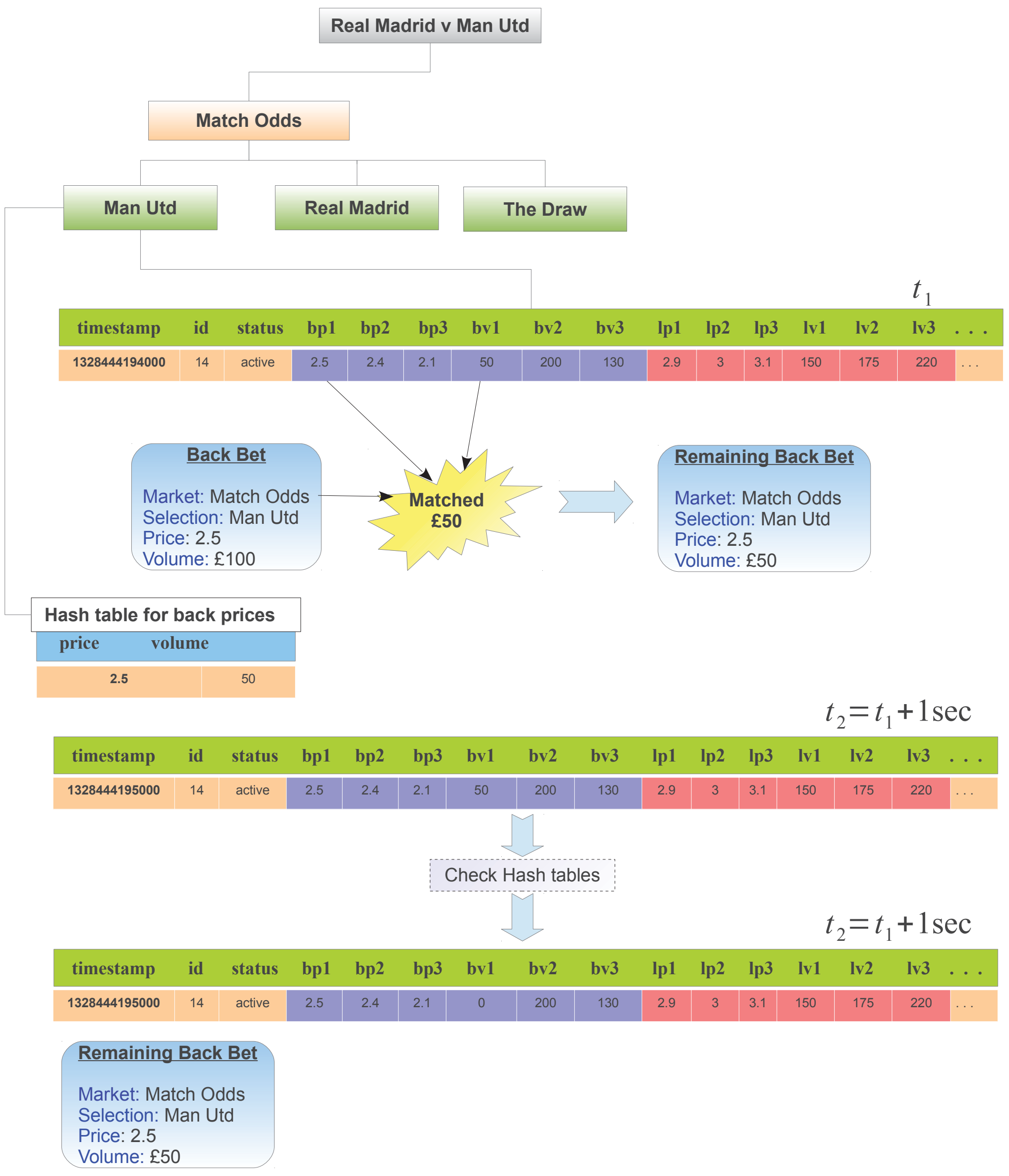

Real Madrid v Man Utd
Match Odds
Man Utd
Real Madrid
The Draw
t_1
timestamp id status bp1 bp2 bp3 bv1 bv2 bv3 lp1 lp2 lp3 lv1 lv2 lv3 . . .
1328444194000 14 active 2.5 2.4 2.1 50 200 130 2.9 3 3.1 150 175 220 . . .
Back Bet
Market: Match Odds
Selection: Man Utd
Price: 2.5
Volume: £100
Matched £50
Remaining Back Bet
Market: Match Odds
Selection: Man Utd
Price: 2.5
Volume: £50
$t_2 = t_1 + 1\text{sec}$
timestamp id status bp1 bp2 bp3 bv1 bv2 bv3 lp1 lp2 lp3 lv1 lv2 lv3 . . .
1328444195000 14 active 2.5 2.4 2.1 50 200 130 2.9 3 3.1 150 175 220 . . .
Remaining Back Bet
Market: Match Odds
Selection: Man Utd
Price: 2.5
Volume: £50
Matched £50

Figure 4.10: The problem of getting matched with already matched volumes

Let us assume that we have the football match Real Madrid v Man Utd and that we wanted to back Man Utd for £100@2.5. Assume that at t_1 our bet can be partially matched for £50. That means there are £50 remaining in our bet. Assume that at t_2 the prices remained the same as in t_1. Because of that we can see that our bet can be matched again for £50. However this is not correct as we have already taken from the market the £50 from t_1. So, there is a need to take into account bets that have already been taken from the user.

SPORTSBET in order to solve this issue is keeping for each selection two hash tables that can map prices to matched volumes. The one hash table is for back prices and the other for lay prices. So, whenever a bet placed from the user get matched, the price and volume that the bet was matched will be stored to the appropriate hash table. Then when new market data arrive SPORTSBET will check if there is any matched volume for the offered prices. If yes then the engine will subtract the matched volume (found in the hash table) from the offered volume and then will update the selection's state (see Figure 4.11).



timestamp	id	status	bp1	bp2	bp3	bv1	bv2	bv3	lp1	lp2	lp3	lv1	lv2	lv3	. . .
1328444194000	14	active	2.5	2.4	2.1	50	200	130	2.9	3	3.1	150	175	220	. . .

price	volume
2.5	50

timestamp	id	status	bp1	bp2	bp3	bv1	bv2	bv3	lp1	lp2	lp3	lv1	lv2	lv3	. . .
1328444195000	14	active	2.5	2.4	2.1	50	200	130	2.9	3	3.1	150	175	220	. . .

timestamp	id	status	bp1	bp2	bp3	bv1	bv2	bv3	lp1	lp2	lp3	lv1	lv2	lv3	. . .
1328444195000	14	active	2.5	2.4	2.1	0	200	130	2.9	3	3.1	150	175	220	. . .

Figure 4.11: SPORTSBET solution for the problem of getting matched with already matched volumes

Chapter 5

SPORTSBET Optimisation Platform

Trading strategies can be specified with a number of tunable parameters (which perhaps for example correspond to thresholds within the strategy), and that clearly it is desirable to have a mechanism in place which can establish (near-)optimal values for these parameters on the basis of an optimisation procedure carried out over historical data. These strategies are becoming more and more complicated and utilise a large amount of data, which makes the back-testing and optimisation process very time consuming. This chapter presents the optimisation platform of SPORTSBET. We present an efficient implementation of the optimisation of trading strategies where strategy parameters are automatically refined using a stochastic search heuristic in order to improve strategy performance and the Walk-Forward Analysis is employed to avoid overfitting. Our implementation can perform the optimisation within a reasonable time range so that the tested trading strategy can be properly deployed in time.

5.1 Introduction

Stochastic optimisation algorithms have been growing rapidly in popularity over the last three decades, with a number of methods now playing a significant role in the analysis, design, and execution of betting strategies. During the optimisation process, a historical simulation (back-testing) will be calculated for a large number of different values of the key strategy parameters so that the optimal values can be identified. In order to do that, all optimisation processes use some type of search method. The methods adopted in stochastic optimisation attempt to model the uncertainty in data by assuming that the input is specified in terms of a probability distribution. Metaheuristics are the most general of this kind of algorithms, and they can be applied to a wide range of problems. The search method will determine the number of back-tests to be performed and therefore the amount of processing time required to complete the process. Moreover, the search method will guide the search in productive directions. However, the directed search methods have some drawbacks. Since a direct search method does not evaluate every possible candidate solution, there is a probability for a certain lack of thoroughness. This can be minimised by selecting the appropriate search method. In order to retrieve from the optimisation process the trading strategy parameters that will most likely produce real-time and long term trading profits, we need to understand the impact of the objective function. The objective function is used during the optimisation process to assign a score in each candidate solution.

5.2 SPORTSBET Search Method

Evolution strategies (ESs) [12] are robust stochastic search algorithms designed to minimise objective functions f that map a continuous search space $\mathbb{R}^n$ into $\mathbb{R}$. An Evolution Strategy is broadly based on the principle of biological evolution. In each generation (iteration) new candidate solutions (denoted as x) are generated by variation, usually in a stochastic way, and then some individuals are selected for the next generation based on their objective function value $f(x)$. Likewise, over the generation sequence, individuals with better and better $f(x)$ are generated. An Evolution Strategy follows these steps [71]:

- **Initialisation**: Many individual solutions are (usually) randomly generated to form an initial population or the initial population can be based upon known good solutions.
- **Evaluation**: The evaluation uses the objective and constraint functions of the optimisation problem to assign a quality score to each individual. Individuals with a higher score

will have a higher probability of surviving and passing their genetic material on (i.e., the candidate solution) to future generations.

- **Selection**: There are two types of selection, the parental selection and the survivor selection. Parental selection is a stochastic selection type that selects the parents that are used for the recombination of a new offspring. In this selection type, the fitter individuals have a higher probability to be selected as parent for recombination. Survivor selection is a deterministic selection that selects the μ fittest individuals either out of the λ offspring (elitist selection) or out of the λ offspring and the μ old parents (non-elitist). This selection type is commonly referred to as $(\mu + \lambda)$ when denoting elitist selection, and as (μ, λ) when denoting non-elitist selection.

- **Mutation and recombination**: Mutation operators add small perturbations to the individuals in the population. Recombination operators recombine two or more individuals in the population into a new individual.

- **Termination**: This generational process is repeated until a termination condition has been reached. Common terminating conditions are: the available computation time, the available number of evaluations/generations and convergence criteria such as a predefined target fitness that is to be reached. After termination, the best solution(s) is found throughout the evolution cycle.

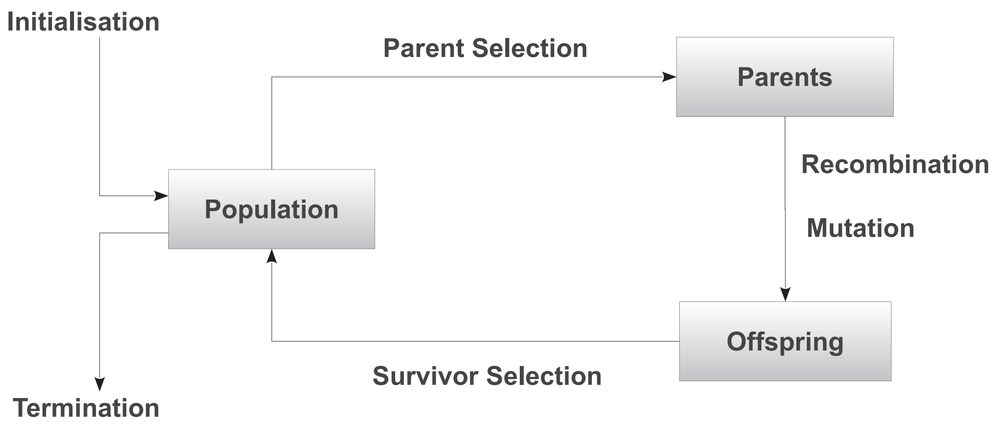


Figure 5.1: Evolutionary Algorithm flowchart

Algorithm 1 The (μ, λ) Evolution Strategy [80]

```
 1: $\mu \leftarrow$ number of parents selected
 2: $\lambda \leftarrow$ number of children generated by the parents
 3: $P \leftarrow \{\}$
 4: for $\lambda$ times do                                          ▷ Build Initial Population
 5:     $P \leftarrow P \cup \{$new random individual$\}$
 6: end for
 7: Best $\leftarrow \Box$
 8: repeat
 9:     for each individual $P_i \in P$ do
10:         AssessFitness $(P_i)$
11:         if Best $= \Box$ or Fitness$(P_i)$ >Fitness(Best) then
12:             Best$\leftarrow P_i$
13:         end if
14:     end for
15:     $Q \leftarrow$ the $\mu$ individuals in $P$ whose Fitness() are greatest      ▷ Truncation Selection
16:     $P \leftarrow \{\}$                          ▷ Join is done by just replacing $P$ with the children
17:     for each individual $Q_j \in Q$ do
18:         for $\lambda \times \mu$ times do
19:             $P \leftarrow P \cup \{Mutate(Copy(Q_j))\}$
20:         end for
21:     end for
22: until Best is the ideal solution or we have run out of time
23: return Best
```

The simplest Evolution Strategy is the (1+1)-ES (one parent, one offspring) [61]. SPORTSBET optimisation platform uses the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [13, 57]. The CMA-ES is a $(\mu/\mu_W, \lambda)$-ES in which all offspring are generated from the same recombinant $\langle x \rangle_W$, computed as the weighted centre of mass of the μ selected individuals, i.e.

$$\langle x \rangle_W = \sum_{i=1}^{\mu} w_i x_{i:\lambda}, \tag{5.1}$$

with $\sum_{i=1}^{\mu} w_i = 1$ and $x_{i:\lambda}$ denoting the object variables of the i^{th} best individual. The offspring are mutated copies of this recombinant, generated as:

$$z_k \sim N(0, I) \tag{5.2}$$

$$y_k = C^{\frac{1}{2}} z \tag{5.3}$$

$$x_k = \langle x \rangle_W + \sigma y_k \tag{5.4}$$

for $k = 1, \ldots, \lambda$. So, the mutations are drawn from a multivariate Gaussian distribution $N(0, \sigma^2 C)$ with C being the covariance matrix of the distribution and σ the step size parameter. The covariance matrix determines the direction of the mutations. It is initialised with $C = I$ and updated each generation based on a weighted empirical covariance estimation of the best individuals of the population (rank-μ-update) and on the direction of the so-called evolution path p_c(rank-one-update). The evolution path (initialized with $p_c = 0$) is defined as the sequence of successive steps taken by the population over a number of generations. The update of the evolution path and the covariance matrix are done in a cumulative way, with

$$\begin{aligned} p_c &= (1 - c_c)p_c + \sqrt{c_c(2 - c_c)\mu_{\mathit{eff}}}\langle y \rangle_W & (5.5)\\ C &= (1 - c_c - c_\mu)C + c_1 p_c p_c^T + c_\mu \sum_{i=1}^{\mu} w_i y_{i:\lambda} & (5.6) \end{aligned}$$

The $\langle y \rangle_W = \sum_{i=1}^{\mu} w_i y_{i:\lambda}$ representing the direction of the step taken by the population's center of mass $\langle x \rangle_W$. The parameter μ_{eff} is the variance effective selection mass. The parameters c, c_c , c_1 are the cumulative factors for the the rank--update, evolution path update, and the rank-one-update respectively. The stepsize parameter σ scales the mutations and it is updated in each generation using an exponential update. The cumulated stepsize adaptation mechanism also uses the evolution path p_σ (initialized with $p_\sigma = 0$), which registers both the length and direction of the evolution path.

$$p_\sigma = (1 - c_\sigma)p_\sigma + \sqrt{c_\sigma(2 - c_\sigma)\mu_{\mathit{eff}}} C^{-\frac{1}{2}} \langle y \rangle_W \tag{5.7}$$

$$\sigma = \sigma \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|p_\sigma\|}{E[\|N(0, I)\|]}\right)\right) \tag{5.8}$$

The setting of the parameters used within the CMA-ES are:

Default population size:

$$\lambda = 4 + \lfloor 3 \ln n \rfloor, \;\; \mu = \lfloor \mu' \rfloor, \;\; \mu' = \frac{\lambda}{2} \tag{5.9}$$

Recombination weights:

$$w_i = \frac{w_i}{\sum_{j=1}^{\mu} w'_j}, \quad w'_j = \ln(\mu' + 0.5) - \ln j, \quad for\ i = 1, ...\mu \tag{5.10}$$

Variance effective selection mass:

$$\mu_{\mathit{eff}} = \left(\sum_{i=1}^{\mu} w_i^2 \right)^{-1} \tag{5.11}$$

Covariance matrix adaptation parameters:

$$c_c = \frac{4 + \frac{\mu_{\mathit{eff}}}{n}}{n + 4 + \frac{2\mu_{\mathit{eff}}}{n}}, \quad c_1 = \frac{2}{(n+1.3)^2 + \mu_{\mathit{eff}}}, \quad c_\mu = \min \left(1 - c_1, 2\frac{\mu_{\mathit{eff}} - 2 + \frac{1}{\mu_{\mathit{eff}}}}{(n+2)^2 + \mu_{\mathit{eff}}} \right) \tag{5.12}$$

Stepsize adaptation parameters:

$$c_\sigma = \frac{\mu_{\mathit{eff}} + 2}{n + \mu_{\mathit{eff}} + 5}, \quad d_\sigma = 1 + 2\max\left(0, \sqrt{\frac{\mu_{\mathit{eff}} - 1}{n+1}} - 1\right) + c_\sigma \tag{5.13}$$

In many real-parameter optimisation problems the search space is a hypercube defined by lower and upper boundary values for each parameter $[x_l, x_u]$. Since mutation can yield solutions that are not within this hyperbox, is desirable to implement a box boundary handling algorithm such that each evaluated solution is guaranteed to lie within the search space. For box-constraint handling, two straightforward methods are to reject mutations that fall outside the box or to apply a cut-off rule forcing mutated offspring back to the nearest point on the constraint boundary. In this work we use the rejection method.

For more details the reader is referred to [13, 71].

5.3 Objective Function

A search method continually accepts or rejects trading strategies in the process of seeking the best parameter set in the least time possible. Thus, it is critical to use a proper objective function, which correctly characterises the quality of a trading strategy. As an example if the optimisation process is using as objective function the highest net profit, caution must be exercised in the event that a large proportion of the profit arises from a single large and likely unrepeatable trade with a favourable outcome. Furthermore, using the highest net profit in isolation completely ignores the question of risk. The strategy with the highest net profit could also have a very large and unacceptable drawdown. Or, the strategy parameters selected may have a very small number of trades, which brings into question the statistical validity of these parameters. All of these criteria are very crucial and cannot be ignored when building the objective function. SPORTSBET's back-testing process returns as its result an array of values representing the evaluation of several objective functions, which the user may use as they are, or combine them to produce a new one. The array contains :

- Profit
- Profit after commission
- Maximum Drawdown (MDD)
- Maximum Run Up (MRU)
- Return On Investment (ROI)
- Risk-Adjusted Rate of Return (RAR)
- Reward to Risk Ratio (RRR)
- Perfect Profit
- Number of winning trades
- Number of losing trades

- Pessimistic return on margin (PROM)

- Strategy Efficiency (SE)

5.4 Walk-Forward Analysis

The primary purpose of a Walk-Forward Analysis is to determine the consistency of a trading strategy's performance. This can be achieved by judging the performance of a trading system exclusively on data which have never been a part of the optimisation process which is a far more reliable measure than performance based only on in-sample simulation. The automatic Walk-Forward Analysis is a system design and validation technique in which you optimise the strategy parameter values on a past segment of market data ("in-sample"), then verify the performance of the strategy by testing it forward in time on data following the optimisation segment ("out-of-sample"). The evaluation of the trading strategy is based on how well it performs on the test data ("out-of-sample"), not the data it was optimised on. At the end of the Walk-Forward Analysis we end up with multiple ("out-of-sample") periods and allows us to see how stable a strategy is over time.

Figure 5.2 illustrates the Walk-Forward Analysis procedure. An optimisation is performed over a longer period (in-sample data), and then the optimised parameter set is tested over a subsequent shorter period (out-of-sample data). The optimisation and testing periods are shifted forward, and the process is repeated until a suitable sample size is achieved. The number of the time segments is an optimisation parameter specified by the user and depends on the amount of the available historical data. There are not correct values for the number of time segments and the ratio between "out-of-sample" and "in-sample" data in order to produce good "out-of-sample" results. However, research suggests to use 10 time windows and a ratio of between 15%-25%.

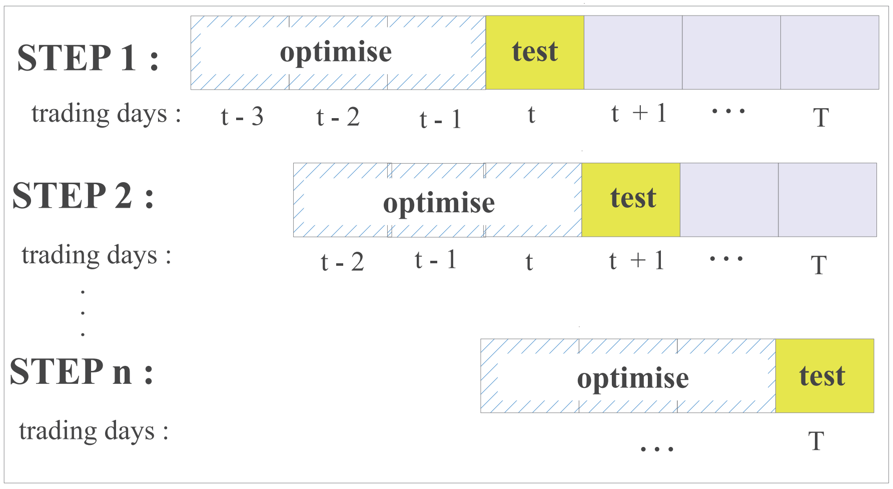


Figure 5.2: Walk-Forward Analysis

In each step the Walk-Forward Efficiency (WFE) is calculated and saved.

$$WFE(\%) = \frac{Annualised\ Net\ Profit\ from\ Testing}{Annualised\ Net\ Profit\ from\ Optimisation} \tag{5.14}$$

The average WFE over the Walk-Forward Analysis can be used to provide some estimation of the rate of profit to be earned during real-time trading. Research has clearly demonstrated that robust trading strategies have WFEs greater than 50-60 percent [93]. Finally another thing to take in consideration is the consistency of the trading strategy. Example:

- **Consistency of profits**: 70 percent of the Walk-Forward windows were profitable.
- **Distribution of profits**: no individual time window contributes more than 50 percent.
- **Maximum Drawdown**: no individual time window had a drawdown of more than 40 percent of initial capital.

Walk-Forward Analysis can be used in a real-time trading system that continues to learn as new data becomes available. Moreover, because we only use the most recent data in each step, we quickly adapt to changes in the market. We get immediate, real-time feedback on the performance of the trading strategy and the current optimal parameter values while the simulation is ongoing. That allows us to spot potential problems with the trading strategy early on.

At the end of the Walk-Forward Analysis we end up with a number of “optimal” solutions. The underlying assumption is that there is no best one for all the tested time segments but rather that the optimal solution changes over time. So those different “optimal” solutions must be examined very careful. If they jump around a lot maybe the trading strategy is not very stable or we might be overfitting. Finally, if the strategy parameters are stable over time we can either use the Sharpe ratio [102] to choose the best one or simply take the mean value in order to come up with a single “optimal” solution.

The Sharpe ratio is a measure in which two measures (mean and variance) can usefully be summarized in one. When the $\overline{WFE}$ is greater than 50-60% and we have to choose among a set of parameters from different time windows, the Sharpe ratio can be used to choose the the set of parameters that are closer to the $\overline{WFE}$.

Let R_S represent the return on set F in the forthcoming period and R_h the return on a historical period. Define d, the differential return, as:

$$d = R_S - R_h \tag{5.15}$$

Let $\overline{d}$ be the expected value of d and σ_d be the predicted standard deviation of d. The Sharpe Ratio S is:

$$S = \frac{\overline{d}}{\sigma_d} \tag{5.16}$$

The CMA-ES is intended really for the case where all the parameters are real-valued. However there are many discrete-valued strategies. We have addressed this issue by introducing a simple heuristic fix by rounding the variable in our fitness evaluation. Which means we do not modify the X-values that CMA-ES is using. The scheme is applicable to any discretized variable. Some more explanation can be found in [58]. In addition we also state that because of the rounding scheme, during the optimisation we might end up with solutions (parameter set) that have already been evaluated. In order to avoid this repetition we keep track of evaluated solutions and if during the optimisation process a parameter set has already been evaluated at a previous step we simply put the result directly in the fitness evaluation.

Chapter 6

SPORTSBET: Third-Party Interface Integration and Implementation Issues

In this chapter, we will consider what betting exchanges API services are, the different services available, and ways of calling those services using UBEL. As an example we will use the Betfair API. Having considered how Betfair API services work, we will explain how SPORTSBET supports those services and adds more functionality to them. Finally, we will discuss some of the basic features of SPORTSBET.

6.1 Exchanges API

The exchange API provides a number of services, which can be called by client programs. Specifically, the exchange API enables developers to create applications that seamlessly integrate with the exchange. Functions, or subroutines in a programming language, may implement those calls in order to get results back from the exchange's servers and process that data, so that it can be used generically within any given program. At a basic level, the website and the exchange API are simply alternate means of accessing the exchange. The exchange that many users access interactively on exchange's website can be accessed programmatically as represented in Figure 6.1.

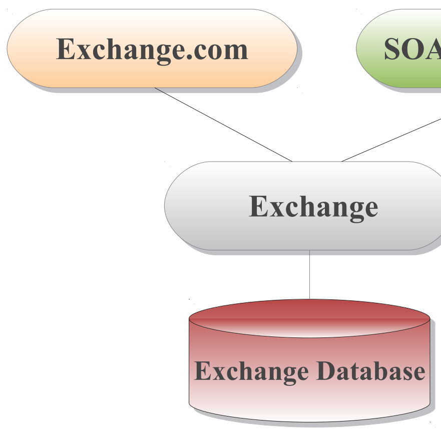


Figure 6.1: Exchange Communication

Description of Betfair API services

Betfair's API enables users to develop applications which integrate with the Betfair sports exchange. The API till now uses a SOAP XML interface to communicate directly with the Betfair database. However, the API from 2013 is changing and using JSON interface for the communication. Using this interface, users can read live market data, place bets and check their account statement without the use of a web browser.

The Betfair API has different service calls available, and levels of availability for those services, depending on the Betfair account holder's subscription to the API. A free access API is offered to all Betfair users which allows them to access the majority of the API's functionality with some limitations at which services are requested. Access to the Free Access API can be enabled in client programs by specifying a product code of "82" to the Login service, together with the Betfair account username and password. The service level for certain calls varies in terms of the number of times per minute that a call to a Free Access API service is made. When a call is restricted this is referred to as "throttling". On top of the limitations posed by the different API packages, Betfair customers are charged a data request fee if a large number of data requests are made within the same second. At the end of every day, Betfair adds up the total number of API and website data requests the user made in each second. If this number exceeds that of 20, the user will be charged an additional 0.1p per data request above 20. API calls and website refreshes send requests for different types and quantities of data. Betfair therefore applies a weighting system to calculate the actual number of data requests made in each second. Table 6.1 shows a list with the most commonly used services available from the Betfair API.

Table 6.1: Betfair API Basic Services

Name	**Description**
Login	Log in to the Exchange API service and initiate a secure session. Users can have multiple sessions alive at any point in time.
Logout	Explicitly end a session.
KeepAlive	Stop a session timing out. Normally a session is expired if it has been idle for a period of time. Issuing periodic KeepAlive requests will stop this occurring.
GetEvents	Navigate through the menu structure. It allows you to input a Sport or Event and retrieve all Events or Markets which have the input event id as a parent.
GetMarket	Retrieve all static market data for a given Market ID.
GetAllMarkets	Retrieve information about all of the markets that are currently active or suspended on the given exchange.
GetMarketPrices	Retrieve dynamic market data for a given Market ID.
GetMarketTradedVolumeCompressed	Obtain the current price (odds) and matched amounts at each price on all of the selections in a particular market.
GetCompleteMarketPricesCompressed	Retrieve all back and lay stakes for each price on the exchange for a given Market ID in a compressed format.
GetCurrentBets	Retrieve information about bets that have been placed by you.
GetBet	Retrieve information for a specific bet that has been placed by you.
PlaceBets	Place multiple bets on a single Market.
UpdateBets	Edit multiple bets on a single Market.
CancelBets	Cancel multiple bets on a single Market.

6.2 SPORTSBET library to call Betfair services

Knowing how the API works in principal, we have a description of the services available, but we have still to make the connection between the Betfair exchange and the client program so that automation takes place. There are many programming languages, such as Java, C++, Python, that have high level user contributed functions and modules to make the task of using web services easier, as well as being able to handle the process from a low level. SPORTSBET using Java to make this connection possible.

Specifically we have created a common Java library which has functions and subroutines to access each of the important services in the Betfair API. For example, the library can be called by other programs in the future. This is far more efficient than writing individual subroutines within individual programs since we will use the service calls again and again and will want to maintain them in one spot. In order to connect UBEL with the Betfair API Java library we have created a Java UObject named `Betfair` which can call all the functions written in the Java library. Moreover, the `Betfair` UObject adds more functionality and more advanced search for specific events and markets. As an example the `Betfair` UObject support filtering in finding the appropriate sporting events and markets in Betfair exchange. Like that someone can search for specific events and markets using filters such as the name of a sporting event, the total ammount of money matched in a market, the number of selections in a market, the market's name, specific selection within a market, if the market is going in-play etc. The available functions of the `Betfair` UObject for searching specific markets and events are:

getLiveFootballEvents, ***getLiveTennisEvents***, ***getLiveBasketballEvents***, ***getLiveFinancialEvents***, ***getLiveHorseRacingEvents***

Input Parameters :

- **selections** : The name of the selections. Ex. "Milan"|"Inter".
- **excludeSelections** : The name of the selections to be excluded.
- **markets** : The name of the market. Ex. "Match Odds"|"Set Betting".

- **excludeMarkets** : The name of the markets to be excluded.

- **countries** : The name of the countries. Ex. "Spain"|"GR".

- **description** : Key words of a sporting event. Ex. "Wimbledon"|"ATP".

- **excludeDescription** : Key words of a sporting event to be excluded.

- **numberOfRunners** : The number of runners in a market. Apply only for Horse Racing. Ex. [5,10] will return markets were $5 \leq number\ of\ runners \leq 10$.

- **numberOfWinners** : The number of winners in a market. Apply only for Horse Racing. Ex. [1,1] will return markets were there is only one winner.

- **totalMatched** : The minimum amount of total matched money in a market.

- **exchangeId** : The id of the exchange (1 = UK, 2 = AUS, 0 = UK and AUS).

- **inplay** : If the market is going in-play (Y = yes, N = no, B = any).

- **days** : The difference from the current time the search will start. Example +1 will start searching for markets 24 hours after the current time.

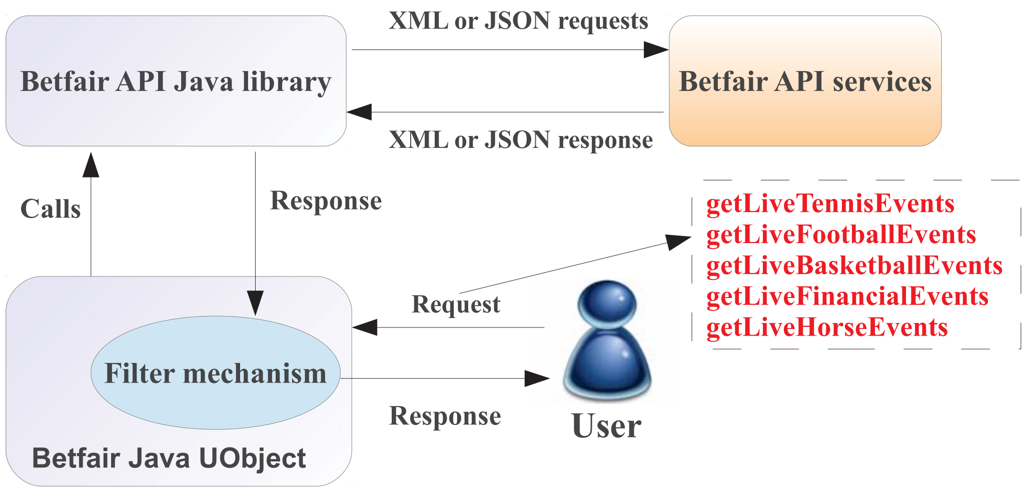


Figure 6.2: SPORTSBET request to Betfair API services

In order to use the Java UObjects you must connect these UObjects remotely to the URBI server. Most live sessions start naturally by logging in to an exchange account via the exchange API. Logging in to the Betfair API is a prerequisite to use any function from Betfair UObject.

Betfair data request charges

As stated before Betfair customers will be charged a data request fee if they make large numbers of data requests within the same second. At the end of every day, Betfair adds up the total number of API and website data requests a user made in each second and if there are more than 20 data requests in one second, Betfair charges the user with 0.1p per data request above 20. API calls and website refreshes send requests for different types and quantities of data [11]. In Figure 6.3 we can see the common way to request market data from the Betfair exchange. As we can see using this type of architecture the total number of requests per second is 12.

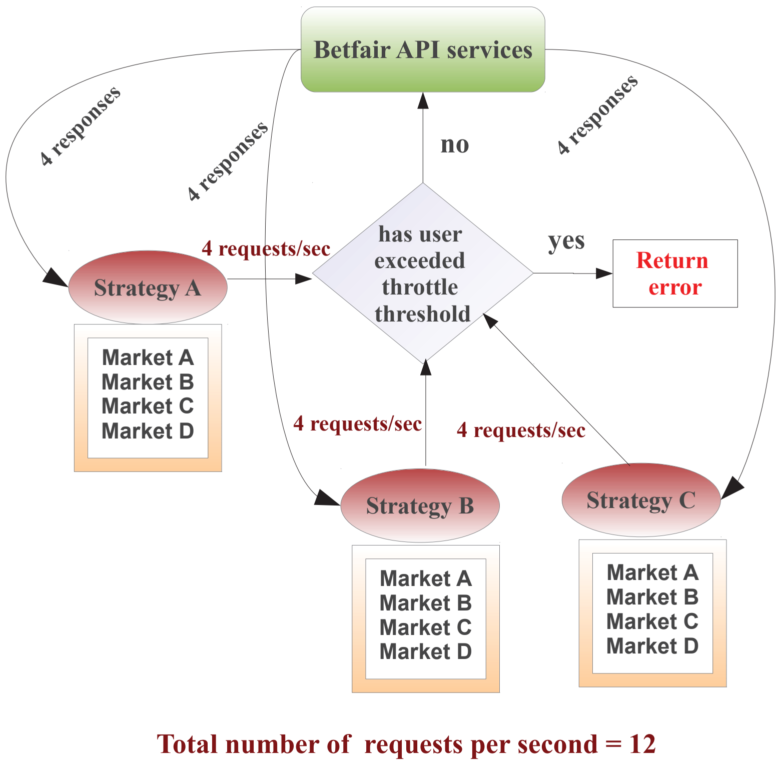


Figure 6.3: Number of requests to Betfair API

SPORTSBET has a request handler which is capable of merging market data requests for the same markets from different strategies. Moreover, it has a response handler which is capable of sending all the market data responses across all the markets of the appropriate strategies. In Figure 6.4 we can see how the total number of data requests per second has been reduced to 4.

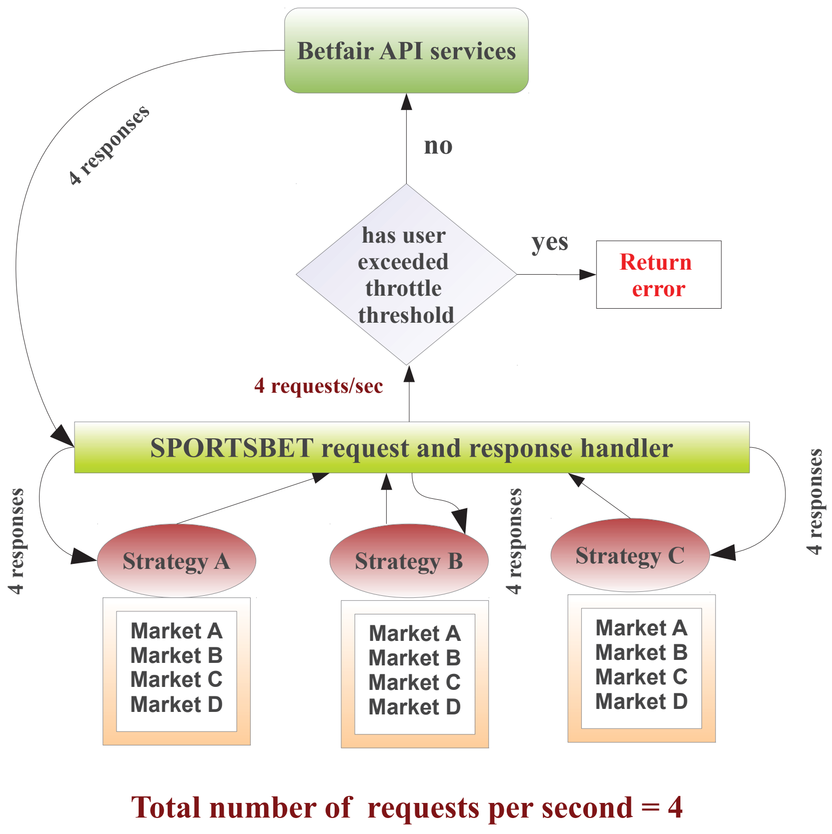


Figure 6.4: Number of requests to Betfair API on SPORTSBET

6.3 Inferring Event Data and Outcome

A trading strategy must be able to react to events taking place within a market (i.e. price movements) and to events occurred within a sporting event (i.e. set finished in a tennis match). The latest usually can be seen from televised broadcasts, web streams or live scoring websites. Ideally, this information should be instant; however, it is not often the case. For instance, television pictures typically transmit with a few seconds delay and therefore bettors with a faster transmission, or present at the game, are able to trade on an informational advantage [16]. Betfair in order to limit this advantage has a delay on placing bets on a market that is in-play[1]. In-play markets usually carry a time delay varying from 1-5 seconds. This delay is in place to allow customers to cancel unmatched orders on the system when there is a change in market conditions.

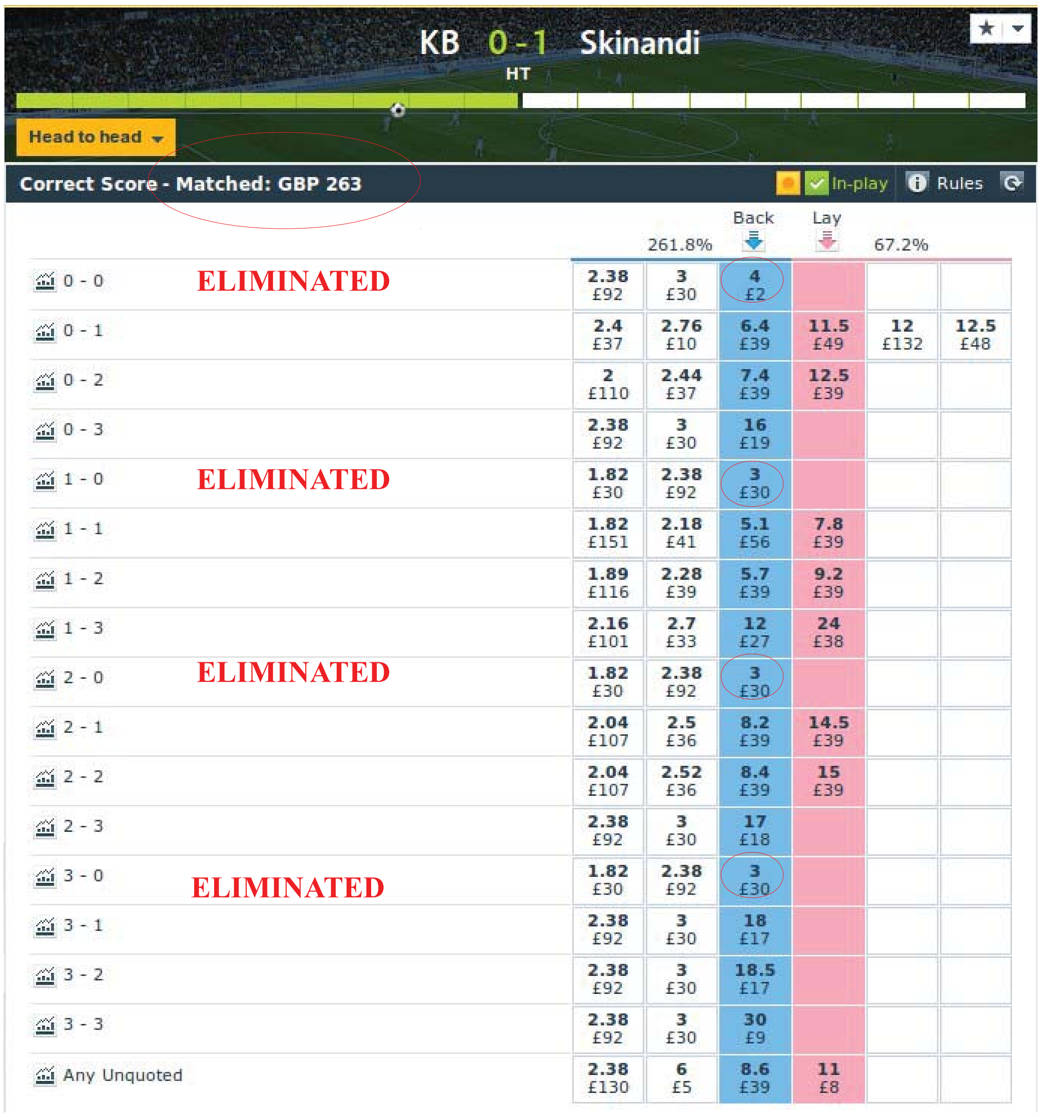
[1]http://help.betfair.info/contents/itemId/i65767339/index.en.html

Currently, the only ways to retrieve events within a sporting event are screen-scraping, scoring websites, which are neither practical nor reliable, or from external feed providers which can be expensive. Because of this, trading bots can only be set to perform trades in response to triggers caused purely by odds movements without taking into account events occurred within a sporting event, as a human would. SPORTSBET's implementation may infer events such as the occurrence of a goal (football) or winning of a set (tennis) from the "Correct Score" and "Set Betting" markets respectively, assuming that the markets are liquid enough.

We will use the example of detecting a goal in football to explain this feature. Each selection in a "Correct Score" market represents a different final score for the football match. Selections will range from "0 - 0", "1 - 0", "0 - 1", all the way up to "3 - 3" and "Any Unquoted". Because this is an in-play market, the odds offered will always be a reflection on the current state of the match. When placing a bet with Betfair, a user may specify odds ranging from 1.01 till 1000. Odds of 1.01 will reflect a 99% chance of the outcome occurring, while odds of 1000 will reflect a 0.001% probability of the outcome occurring. When applied to the "Correct Score" market, odds of 1.01 are normally offered on the current score when there is only one minute remaining in the match. When a market has a lot of liquidity, odds of 1000 will always be placed on all outcomes which are no longer feasible, for example the "2 - 0" outcome on a match which is currently "1 - 1". However, there are cases where the price of no longer feasible outcomes will never be 1000 or will be with a significant delay (see Figure 6.5).

Assuming the market is liquid enough, by analysing at runtime the odds of each outcome, we can accurately infer the current score of the match. We have abstracted this notion further in order to determine whether a goal has been scored, and we do so by comparing the current score with the last score recorded.

This same functionality can be applied to Tennis to detect set winners but cannot be applied to detect the current score in a tennis match. However, a recent research showed that by existing quantitative tennis models it is possible to infer the score of a tennis match solely from live market data of a betting exchange, assuming it has enough liquidity [46].



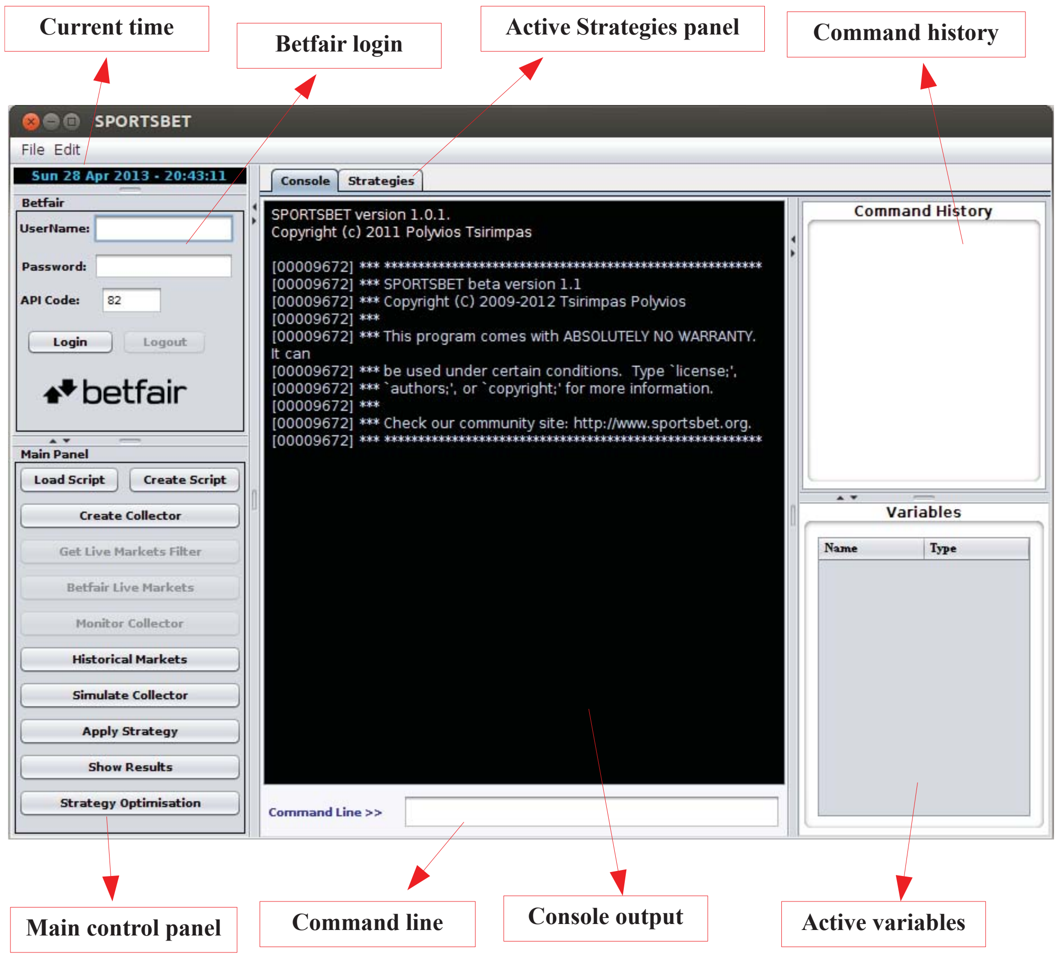
Correct Score - Matched: GBP 263

		261.8%	Back	Lay	67.2%	
0 - 0 ELIMINATED	2.38 £92	3 £30	4 £2			
0 - 1	2.4 £37	2.76 £10	6.4 £39	11.5 £49	12 £132	12.5 £48
0 - 2	2 £110	2.44 £37	7.4 £39	12.5 £39		
0 - 3	2.38 £92	3 £30	16 £19			
1 - 0 ELIMINATED	1.82 £30	2.38 £92	3 £30			
1 - 1	1.82 £151	2.18 £41	5.1 £56	7.8 £39		
1 - 2	1.89 £116	2.28 £39	5.7 £39	9.2 £39		
1 - 3	2.16 £101	2.7 £33	12 £27	24 £38		
2 - 0 ELIMINATED	1.82 £30	2.38 £92	3 £30			
2 - 1	2.04 £107	2.5 £36	8.2 £39	14.5 £39		
2 - 2	2.04 £107	2.52 £36	8.4 £39	15 £39		
2 - 3	2.38 £92	3 £30	17 £18			
3 - 0 ELIMINATED	1.82 £30	2.38 £92	3 £30			
3 - 1	2.38 £92	3 £30	18 £17			
3 - 2	2.38 £92	3 £30	18.5 £17			
3 - 3	2.38 £92	3 £30	30 £9			
Any Unquoted	2.38 £130	6 £5	8.6 £39	11 £8		

Figure 6.5: Example of no goal detection because of low market liquidity

6.4 SPORTSBET's User Interface

As mentioned before SPORTSBET comes with a simple, intuitive and responsive user interface, ultimately minimising the effort and time consumed to perform a desired task. In Figure 6.6 we can see the main view of SPORTSBET's GUI.



```
SPORTSBET version 1.0.1.
Copyright (c) 2011 Polyvios Tsirimpas

[00009672] *** ******************************************************
[00009672] *** SPORTSBET beta version 1.1
[00009672] *** Copyright (C) 2009-2012 Tsirimpas Polyvios
[00009672] ***
[00009672] *** This program comes with ABSOLUTELY NO WARRANTY.
It can
[00009672] *** be used under certain conditions.  Type `license;',
[00009672] *** `authors;', or `copyright;' for more information.
[00009672] ***
[00009672] *** Check our community site: http://www.sportsbet.org.
[00009672] *** ******************************************************
```

Figure 6.6: SPORTSBET's user interface

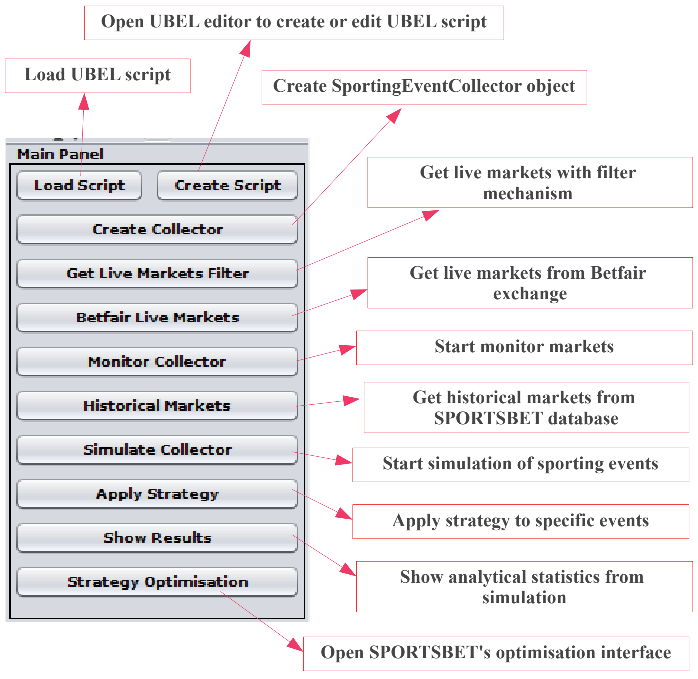

Open UBEL editor to create or edit UBEL script
Load UBEL script
Create SportingEventCollector object
Main Panel
Load Script
Create Script
Create Collector
Get Live Markets Filter
Betfair Live Markets
Monitor Collector
Historical Markets
Simulate Collector
Apply Strategy
Show Results
Strategy Optimisation
Get live markets with filter mechanism
Get live markets from Betfair exchange
Start monitor markets
Get historical markets from SPORTSBET database
Start simulation of sporting events
Apply strategy to specific events
Show analytical statistics from simulation
Open SPORTSBET's optimisation interface

Figure 6.7: SPORTSBET's main control panel

- **Get Live Markets Filter** : A graphical interface to filter requested events and markets from betting exchange platforms. (see Figure 6.8).

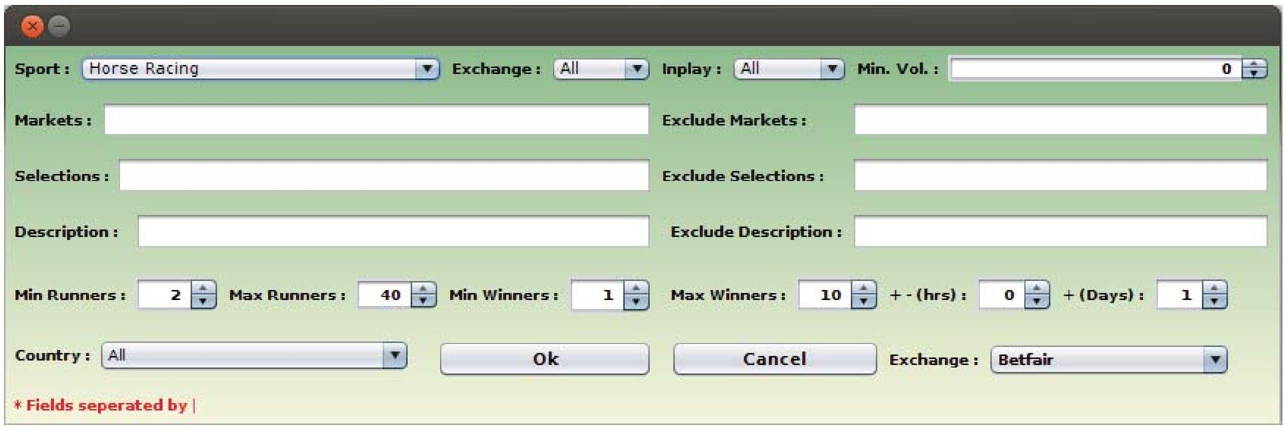
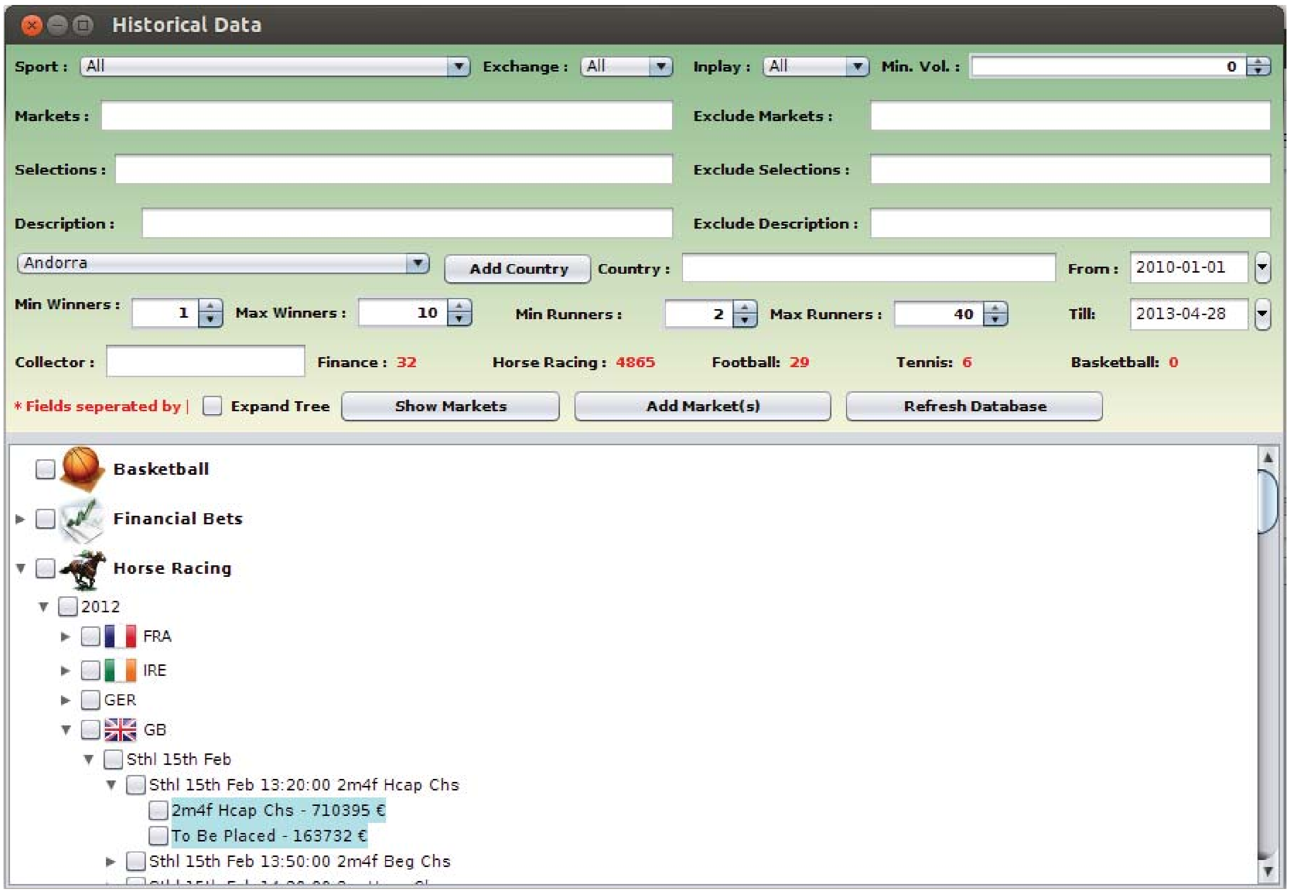

Sport : Horse Racing
Exchange : All
Inplay : All
Min. Vol. : 0
Markets :
Exclude Markets :
Selections :
Exclude Selections :
Description :
Exclude Description :
Min Runners : 2
Max Runners : 40
Min Winners : 1
Max Winners : 10
+ - (hrs) : 0
+ (Days) : 1
Country : All
Exchange : Betfair
* Fields seperated by |

Figure 6.8: Filter mechanism for live markets

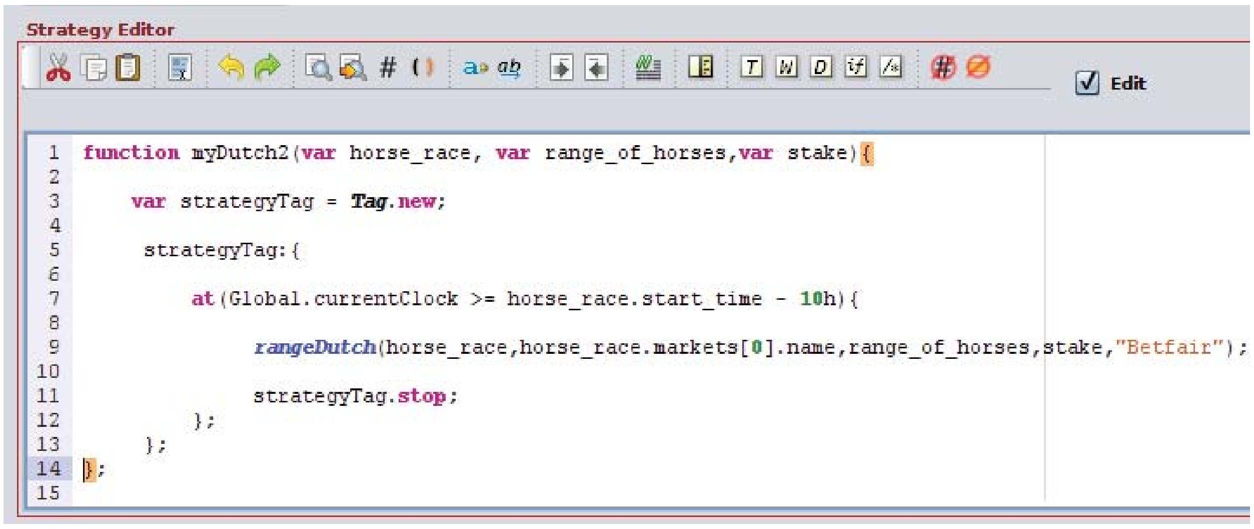
- **Get Historical Markets Filter** : A graphical interface to filter requested events from SPORTSBET database. (see Figure 6.9).


Historical Data
Sport : All
Exchange : All
Inplay : All
Min. Vol. : 0
Markets :
Exclude Markets :
Selections :
Exclude Selections :
Description :
Exclude Description :
Andorra
Country :
From : 2010-01-01
Min Winners : 1
Max Winners : 10
Min Runners : 2
Max Runners : 40
Till: 2013-04-28
Collector :
Finance : 32
Horse Racing : 4865
Football: 29
Tennis: 6
Basketball: 0
* Fields seperated by |
Expand Tree

Basketball
Financial Bets
Horse Racing
2012
FRA
IRE
GER
GB
Sthl 15th Feb
Sthl 15th Feb 13:20:00 2m4f Hcap Chs
2m4f Hcap Chs - 710395 €
To Be Placed - 163732 €
Sthl 15th Feb 13:50:00 2m4f Beg Chs

Figure 6.9: Filter mechanism for historical markets

- **UBELtext** : A text editor for the UBEL (see Figure 6.10).

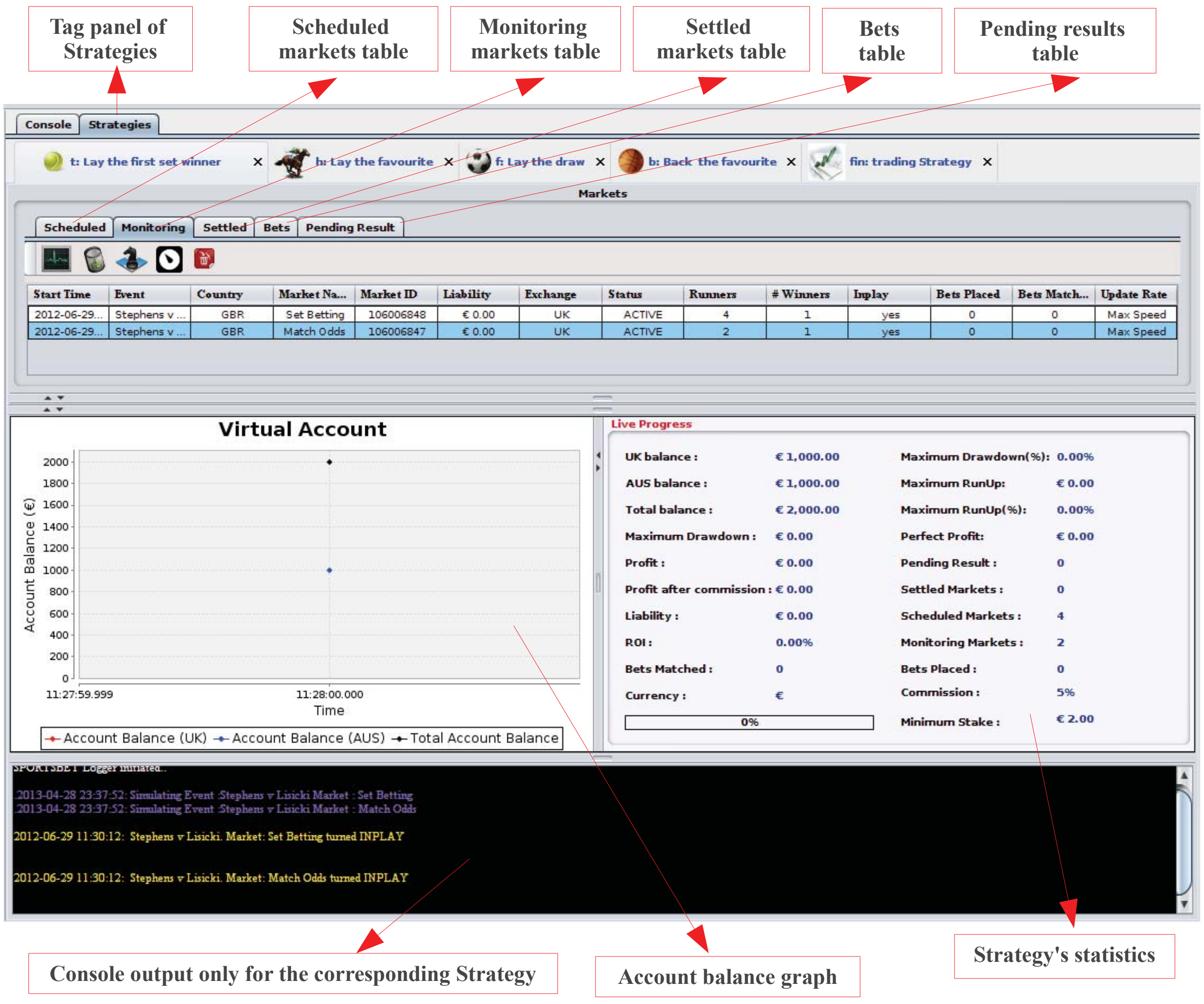


```
function myDutch2(var horse_race, var range_of_horses,var stake){

    var strategyTag = Tag.new;

     strategyTag:{

         at(Global.currentClock >= horse_race.start_time - 10h){

              rangeDutch(horse_race,horse_race.markets[0].name,range_of_horses,stake,"Betfair");

              strategyTag.stop;
         };
     };
};
```

Figure 6.10: UBELtext

UBELtext is written in Java, so it has the ability to run in all operating systems, and was designed to help UBEL programmers to write UBEL code faster and easier. UBEL-text supports auto indent, syntax highlighting, and every other feature, both basic and advanced, you expect to find in a text editor.

- **Strategy viewer** : A graphical view of the active strategies. Each strategy panel, has five tables showing the scheduled markets of the strategy, the ones currently testing/-monitoring, the settled markets, the bets placed and the markets closed but pending the result. Moreover, the strategy panel consists of a Market viewer area, a strategy statistics and graph area, and finally a console area where someone can see the activity of the specific strategy (see Figure 6.11).



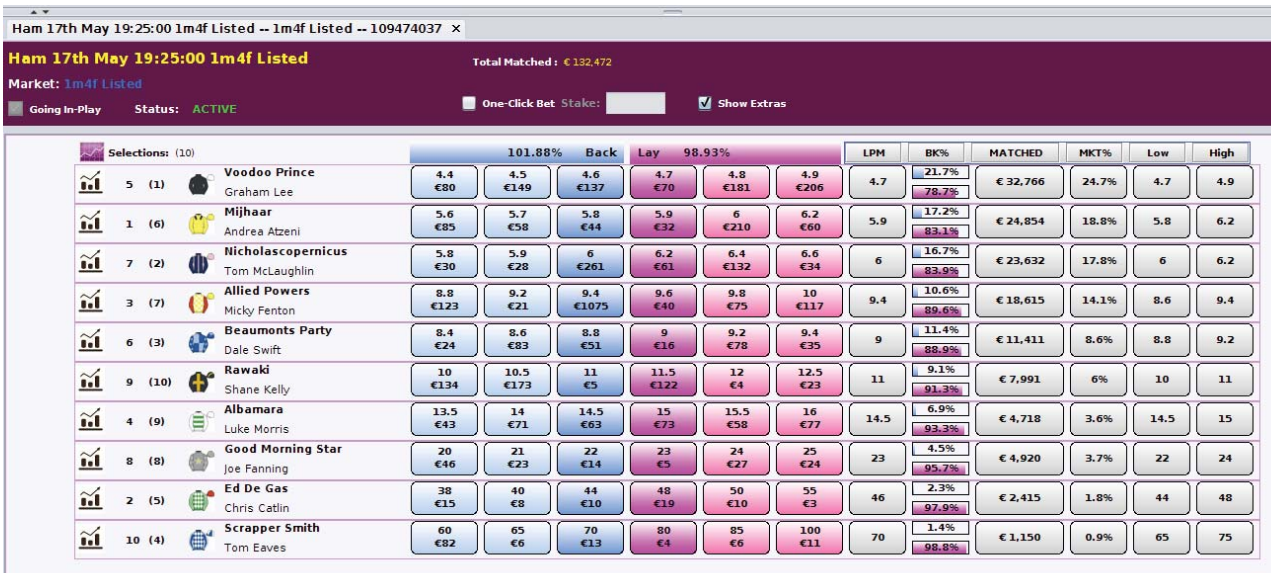
Start Time	Event	Country	Market Na...	Market ID	Liability	Exchange	Status	Runners	# Winners	Inplay	Bets Placed	Bets Match...	Update Rate
2012-06-29...	Stephens v ...	GBR	Set Betting	106006848	€ 0.00	UK	ACTIVE	4	1	yes	0	0	Max Speed
2012-06-29...	Stephens v ...	GBR	Match Odds	106006847	€ 0.00	UK	ACTIVE	2	1	yes	0	0	Max Speed

UK balance :	€ 1,000.00	Maximum Drawdown(%):	0.00%
AUS balance :	€ 1,000.00	Maximum RunUp:	€ 0.00
Total balance :	€ 2,000.00	Maximum RunUp(%):	0.00%
Maximum Drawdown :	€ 0.00	Perfect Profit:	€ 0.00
Profit :	€ 0.00	Pending Result :	0
Profit after commission :	€ 0.00	Settled Markets :	0
Liability :	€ 0.00	Scheduled Markets :	4
ROI :	0.00%	Monitoring Markets :	2
Bets Matched :	0	Bets Placed :	0
Currency :	€	Commission :	5%
0%		Minimum Stake :	€ 2.00

```
SPORTSBET Logger initiated..

.2013-04-28 23:37:52: Simulating Event :Stephens v Lisicki Market : Set Betting
.2013-04-28 23:37:52: Simulating Event :Stephens v Lisicki Market : Match Odds

2012-06-29 11:30:12:  Stephens v Lisicki. Market: Set Betting turned INPLAY


2012-06-29 11:30:12:  Stephens v Lisicki. Market: Match Odds turned INPLAY
```

Figure 6.11: Strategy Viewer

- **Market viewer** : Provides similar view of a market with the one shown on the betting exchange web page. Moreover, it shows more statistics for each selection such as, the last matched price, the matched volume, the percentage of the matched volume to the market total matched volume, the highest and lowest price over the monitoring time period (see Figure 6.12). The market viewer comes with advanced charting facilities (see Figures 6.13, 6.14) where someone can see the visualisation of odds evolution for each selection in a market.



Ham 17th May 19:25:00 1m4f Listed -- 1m4f Listed -- 109474037 ×

Ham 17th May 19:25:00 1m4f Listed

Total Matched : € 132,472

Market: 1m4f Listed

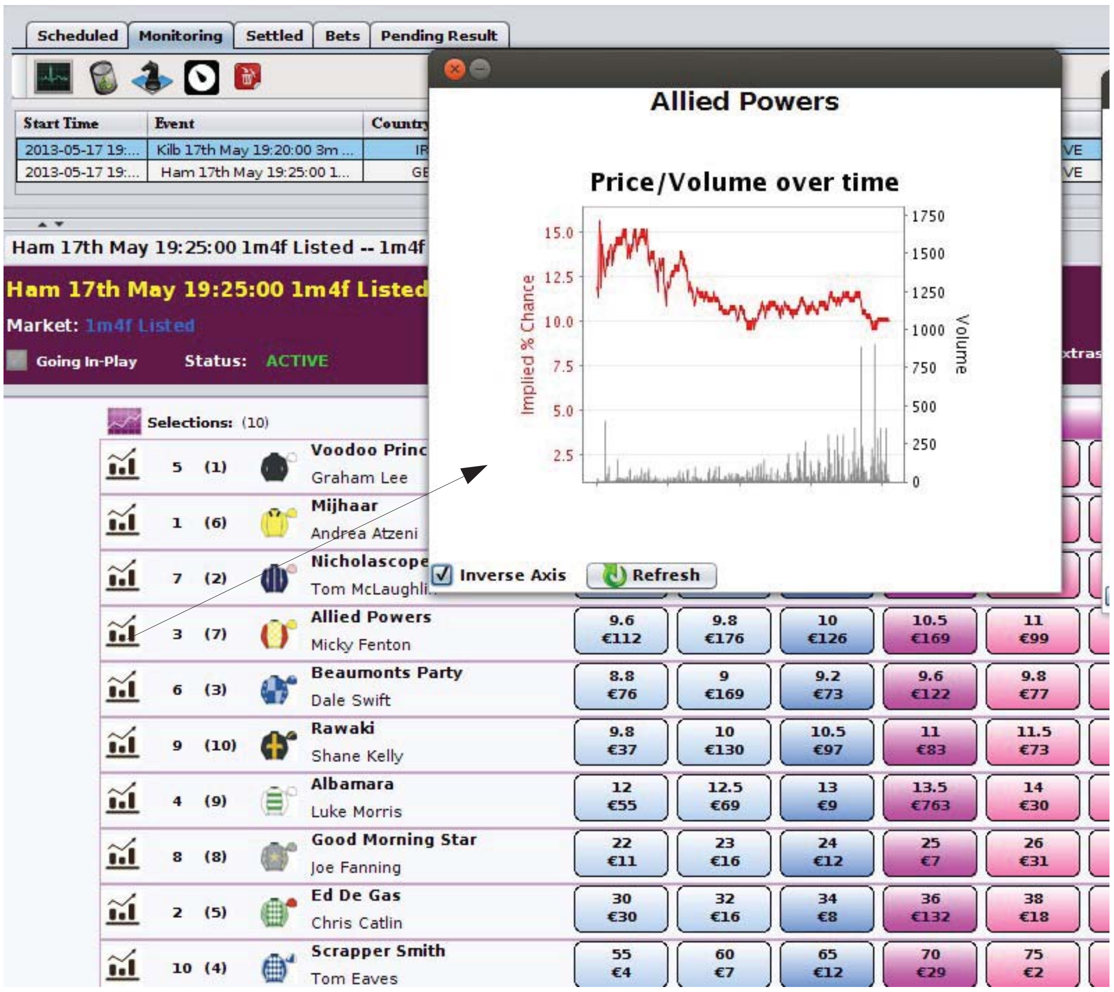
Going In-Play Status: ACTIVE

One-Click Bet Stake: ☑ Show Extras

Selections: (10)

				101.88% Back			Lay 98.93%			LPM	BK%	MATCHED	MKT%	Low	High
5	(1)	Voodoo Prince	Graham Lee	4.4 €80	4.5 €149	4.6 €137	4.7 €70	4.8 €181	4.9 €206	4.7	21.7% 78.7%	€ 32,766	24.7%	4.7	4.9
1	(6)	Mijhaar	Andrea Atzeni	5.6 €85	5.7 €58	5.8 €44	5.9 €32	6 €210	6.2 €60	5.9	17.2% 83.1%	€ 24,854	18.8%	5.8	6.2
7	(2)	Nicholascopernicus	Tom McLaughlin	5.8 €30	5.9 €28	6 €261	6.2 €61	6.4 €132	6.6 €34	6	16.7% 83.9%	€ 23,632	17.8%	6	6.2
3	(7)	Allied Powers	Micky Fenton	8.8 €123	9.2 €21	9.4 €1075	9.6 €40	9.8 €75	10 €117	9.4	10.6% 89.6%	€ 18,615	14.1%	8.6	9.4
6	(3)	Beaumonts Party	Dale Swift	8.4 €24	8.6 €83	8.8 €51	9 €16	9.2 €78	9.4 €35	9	11.4% 88.9%	€ 11,411	8.6%	8.8	9.2
9	(10)	Rawaki	Shane Kelly	10 €134	10.5 €173	11 €5	11.5 €122	12 €4	12.5 €23	11	9.1% 91.3%	€ 7,991	6%	10	11
4	(9)	Albamara	Luke Morris	13.5 €43	14 €71	14.5 €63	15 €73	15.5 €58	16 €77	14.5	6.9% 93.3%	€ 4,718	3.6%	14.5	15
8	(8)	Good Morning Star	Joe Fanning	20 €46	21 €23	22 €14	23 €5	24 €27	25 €24	23	4.5% 95.7%	€ 4,920	3.7%	22	24
2	(5)	Ed De Gas	Chris Catlin	38 €15	40 €8	44 €10	48 €19	50 €10	55 €3	46	2.3% 97.9%	€ 2,415	1.8%	44	48
10	(4)	Scrapper Smith	Tom Eaves	60 €82	65 €6	70 €13	80 €4	85 €6	100 €11	70	1.4% 98.8%	€ 1,150	0.9%	65	75

Figure 6.12: Market Viewer



Start Time	Event	Country
2013-05-17 19:...	Kilb 17th May 19:20:00 3m ...	IR
2013-05-17 19:...	Ham 17th May 19:25:00 1...	GE

Ham 17th May 19:25:00 1m4f Listed -- 1m4f

Ham 17th May 19:25:00 1m4f Listed

Market: 1m4f Listed

Going In-Play Status: ACTIVE

Selections: (10)

5	(1)	Voodoo Princ	Graham Lee					
1	(6)	Mijhaar	Andrea Atzeni					
7	(2)	Nicholascope	Tom McLaughli					
3	(7)	Allied Powers	Micky Fenton	9.6 €112	9.8 €176	10 €126	10.5 €169	11 €99
6	(3)	Beaumonts Party	Dale Swift	8.8 €76	9 €169	9.2 €73	9.6 €122	9.8 €77
9	(10)	Rawaki	Shane Kelly	9.8 €37	10 €130	10.5 €97	11 €83	11.5 €73
4	(9)	Albamara	Luke Morris	12 €55	12.5 €69	13 €9	13.5 €763	14 €30
8	(8)	Good Morning Star	Joe Fanning	22 €11	23 €16	24 €12	25 €7	26 €31
2	(5)	Ed De Gas	Chris Catlin	30 €30	32 €16	34 €8	36 €132	38 €18
10	(4)	Scrapper Smith	Tom Eaves	55 €4	60 €7	65 €12	70 €29	75 €2

Figure 6.13: Visualisation of odds evolution over the course of a single selection (Allied Power), from web page.

In Figure 6.14 the blue line represents the % implied chance of the best available price to back, the pink one represents the % implied chance of the best available price to lay while the green one represents the last price matched value.

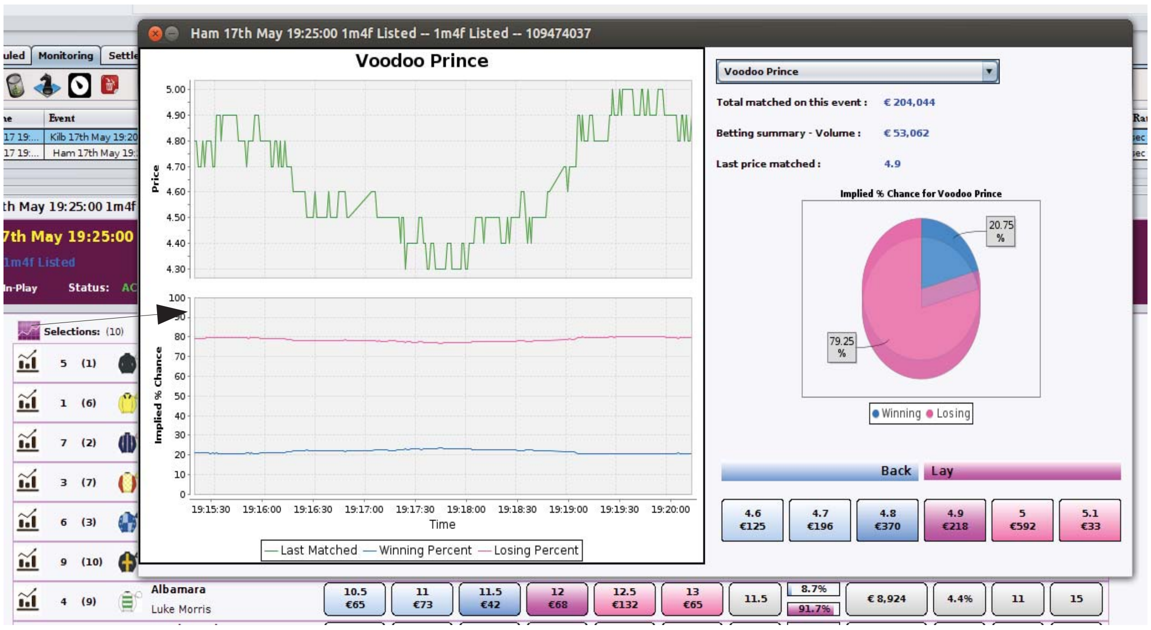


Figure 6.14: SPORTSBET's visualisation of odds evolution over the course of a single selection (Voodoo Prince).

SPORTSBET is the only tool currently available that offers a visualisation of a strategy combined with the odds evolution. This gives greater insight into strategy operation. Specifically, the user can see on the graph:

– Events: For example, a goal being scored in a football match, the end of a set in a tennis match, the end of half time in a basketball or football match. These events are shown on the graph as vertical lines and there is a text next to them showing the event. see (Figures 6.15, 6.16).

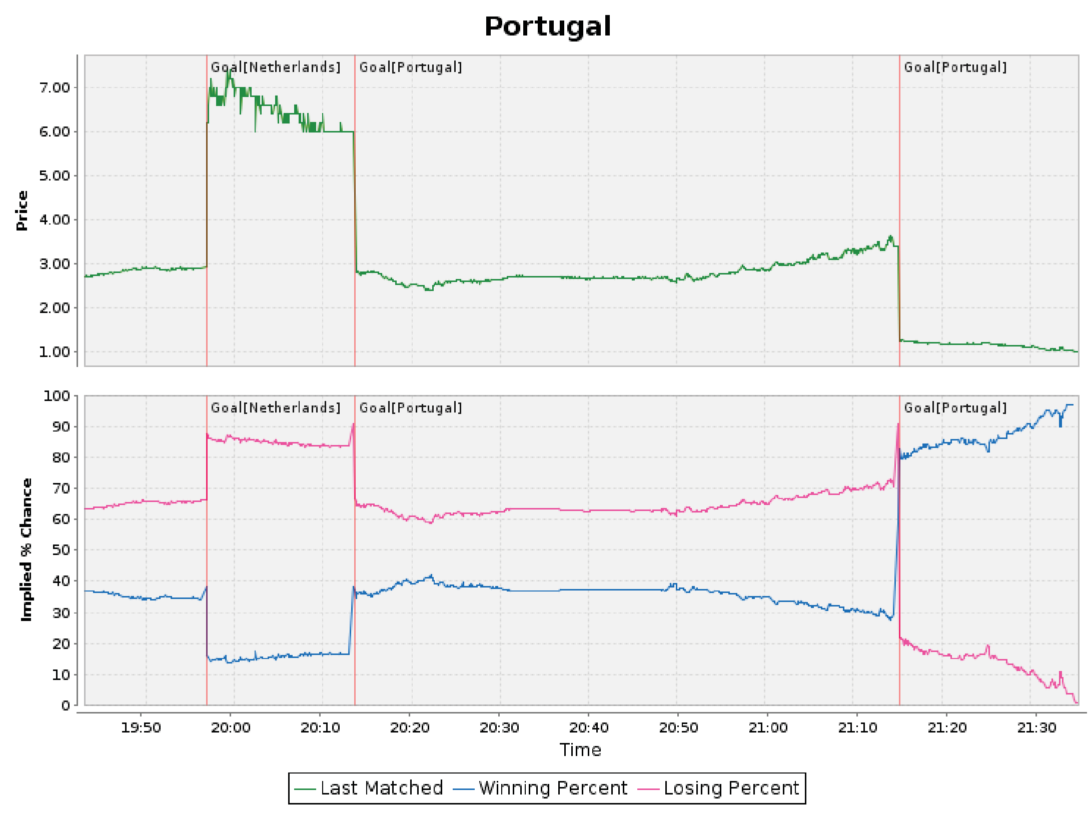


Figure 6.15: Event visualisation – Goals being scored

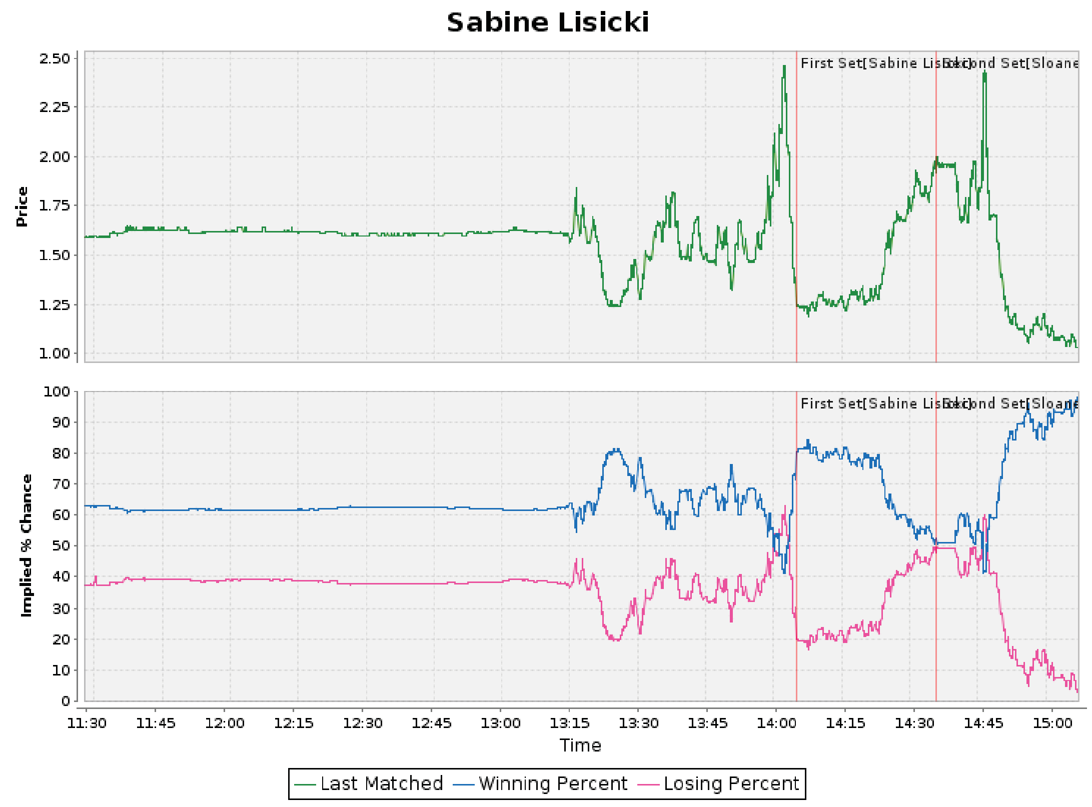


Figure 6.16: Event visualisation – Set finished

– Bets: Unmatched bets appear as white boxes with blue outline for back bets and pink for lay bets. When the bet is matched a line connecting the unmatched bet with the matched bet appears and the matched bet will be proportionally filled with the volume matched. Moreover, someone can see when the price or volume of a bet is changed or a bet is cancelled (see Figures 6.17, 6.18, 6.19).

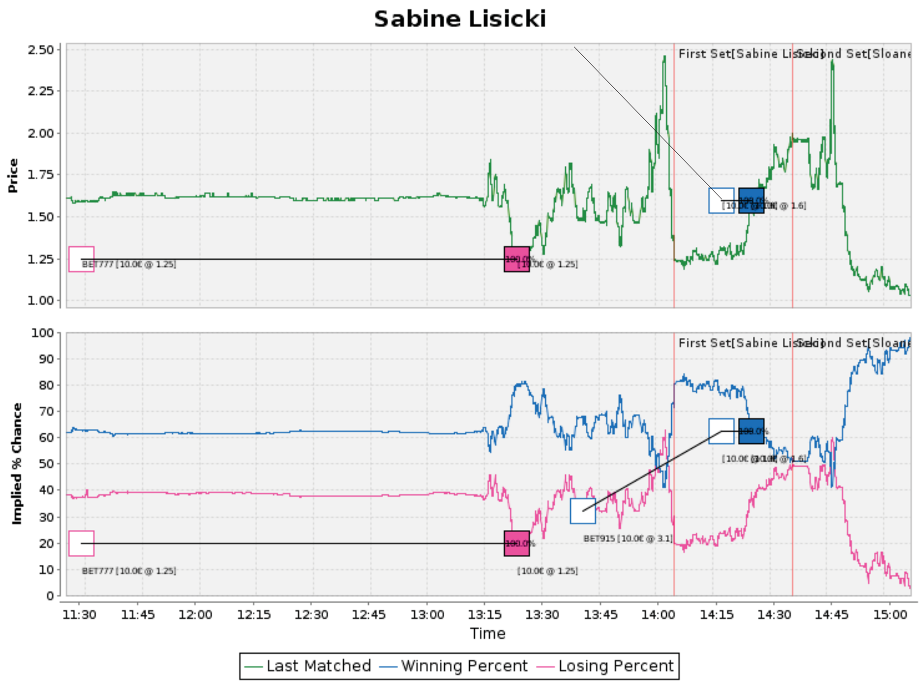


Figure 6.17: Bets visualisation: Lay bet placed and matched. Back bet placed, back bet price changed and finally back bet matched.

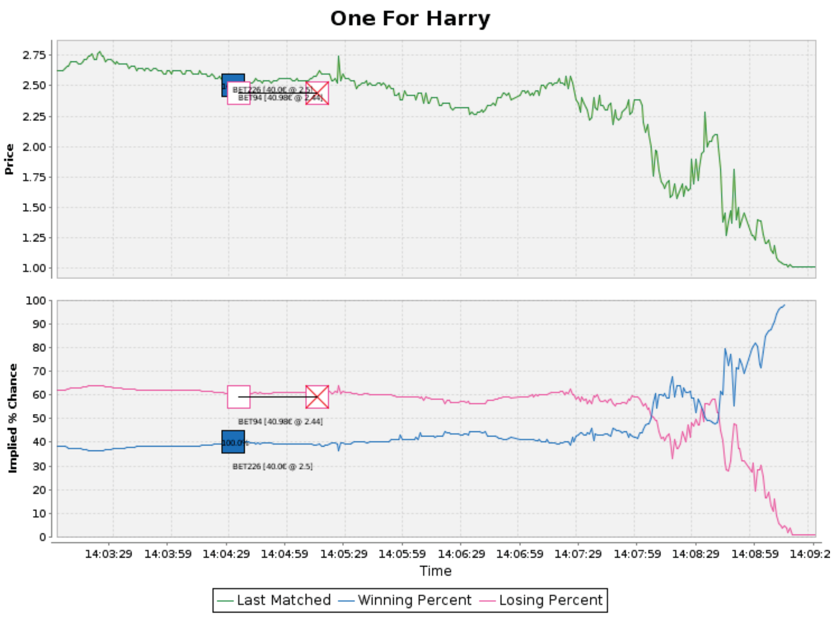


Figure 6.18: Bets visualisation: Back bet matched at the best available odds. Lay bet cancelled.

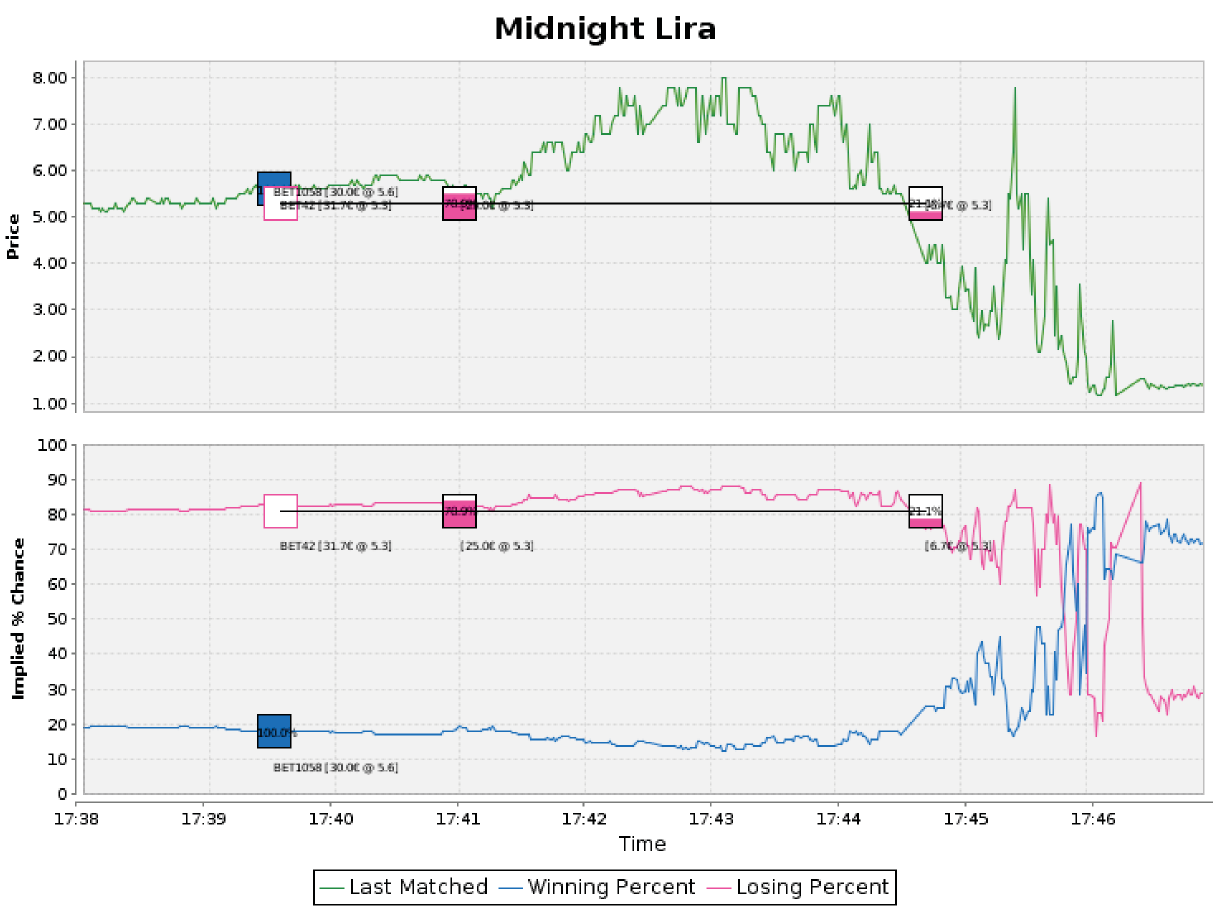
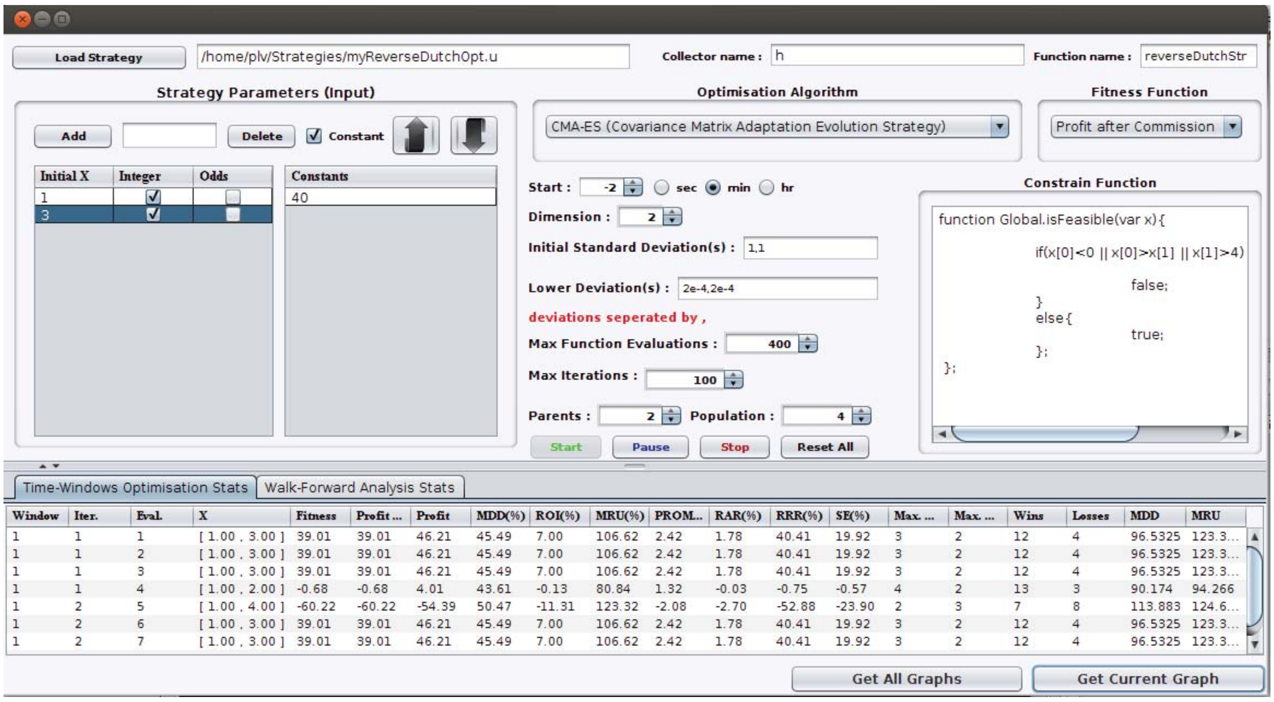


Figure 6.19: Bets visualisation: Back bet placed and matched at the best available odds. Lay bet placed, lay bet partially matched (78.9%) and finally remaining volume matched (21.1%).

- **Optimisation Tool Interface**: A graphical tool interface that allows the user to quickly set-up, run and evaluate a strategy optimisation problem. Using the optimisation tool the user can define, solve and access the optimisation problems. The user can choose the optimisation settings, such as the metaheuristic algorithm, the objective function, the dimension of the problem, the initial standard deviation of their startegy's parameters, the maximum number of iterations and evaluations, etc. The user can also define the constrain function of the optimisation problem. Moreover, the user can see the progress of the optimisation as text or the visualisation of it on a graph (see Figures 6.20, 6.21).



/home/plv/Strategies/myReverseDutchOpt.u

Collector name : h

Function name : reverseDutchStr

Strategy Parameters (Input)

✓ Constant

Initial X	Integer	Odds
1	✓	
3	✓	

Constants
40

Optimisation Algorithm

CMA-ES (Covariance Matrix Adaptation Evolution Strategy)

Fitness Function

Profit after Commission

Start : -2 sec [✓ min] hr

Dimension : 2

Initial Standard Deviation(s) : 1,1

Lower Deviation(s) : 2e-4,2e-4

deviations seperated by ,

Max Function Evaluations : 400

Max Iterations : 100

Parents : 2 Population : 4

Constrain Function

```
function Global.isFeasible(var x){
        if(x[0]<0 || x[0]>x[1] || x[1]>4)
                false;
        }
        else{
                true;
        };
};
```

Time-Windows Optimisation Stats | Walk-Forward Analysis Stats

Window	Iter.	Eval.	X	Fitness	Profit ...	Profit	MDD(%)	ROI(%)	MRU(%)	PROM..	RAR(%)	RRR(%)	SE(%)	Max. ...	Max. ...	Wins	Losses	MDD	MRU
1	1	1	[1.00 , 3.00]	39.01	39.01	46.21	45.49	7.00	106.62	2.42	1.78	40.41	19.92	3	2	12	4	96.5325	123.3...
1	1	2	[1.00 , 3.00]	39.01	39.01	46.21	45.49	7.00	106.62	2.42	1.78	40.41	19.92	3	2	12	4	96.5325	123.3...
1	1	3	[1.00 , 3.00]	39.01	39.01	46.21	45.49	7.00	106.62	2.42	1.78	40.41	19.92	3	2	12	4	96.5325	123.3...
1	1	4	[1.00 , 2.00]	-0.68	-0.68	4.01	43.61	-0.13	80.84	1.32	-0.03	-0.75	-0.57	4	2	13	3	90.174	94.266
1	2	5	[1.00 , 4.00]	-60.22	-60.22	-54.39	50.47	-11.31	123.32	-2.08	-2.70	-52.88	-23.90	2	3	7	8	113.883	124.6...
1	2	6	[1.00 , 3.00]	39.01	39.01	46.21	45.49	7.00	106.62	2.42	1.78	40.41	19.92	3	2	12	4	96.5325	123.3...
1	2	7	[1.00 , 3.00]	39.01	39.01	46.21	45.49	7.00	106.62	2.42	1.78	40.41	19.92	3	2	12	4	96.5325	123.3...

Figure 6.20: Optimisation Platform

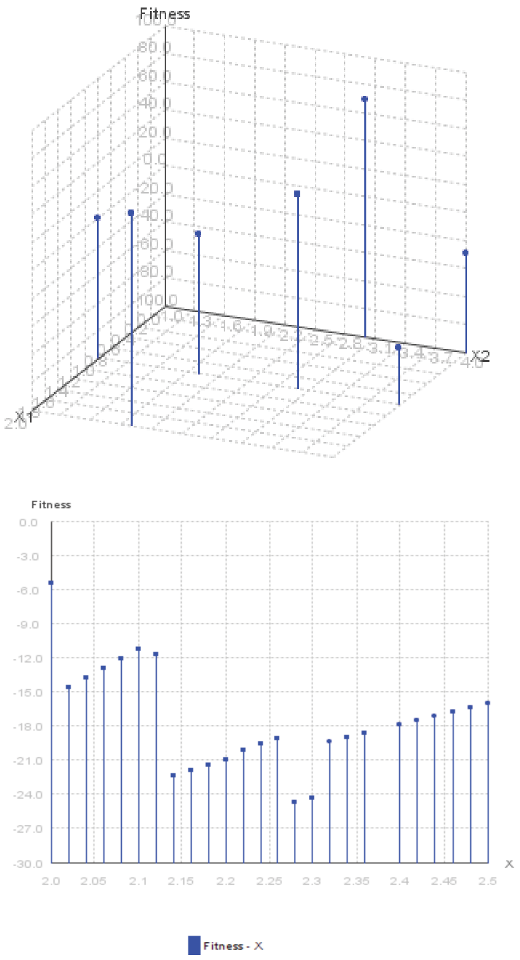
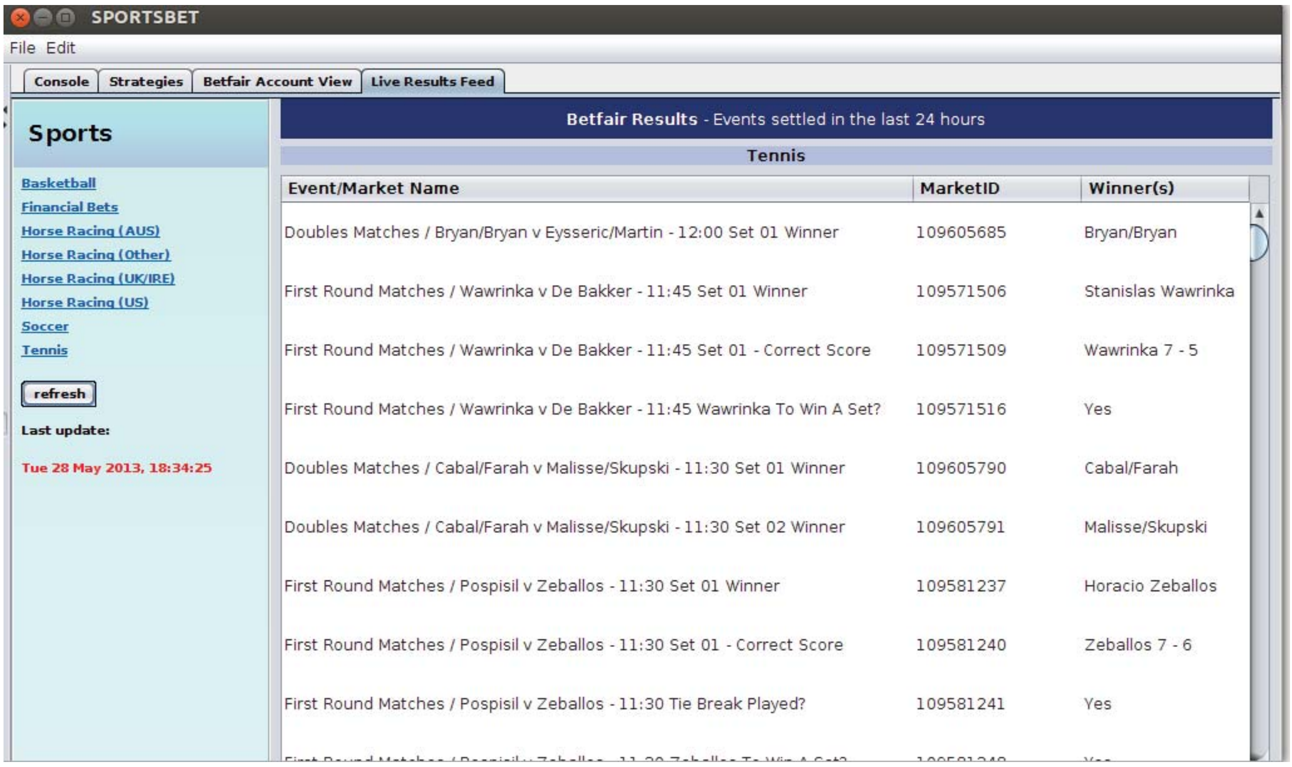


Figure 6.21: Optimisation progress graphical view

- **RssFeed Results**: A module that allows the user to see results for all the sporting events and their associated markets from Betfair. (see Figure 6.22).



Last update:

Tue 28 May 2013, 18:34:25

Betfair Results - Events settled in the last 24 hours

Tennis

Event/Market Name	MarketID	Winner(s)
Doubles Matches / Bryan/Bryan v Eysseric/Martin - 12:00 Set 01 Winner	109605685	Bryan/Bryan
First Round Matches / Wawrinka v De Bakker - 11:45 Set 01 Winner	109571506	Stanislas Wawrinka
First Round Matches / Wawrinka v De Bakker - 11:45 Set 01 - Correct Score	109571509	Wawrinka 7 - 5
First Round Matches / Wawrinka v De Bakker - 11:45 Wawrinka To Win A Set?	109571516	Yes
Doubles Matches / Cabal/Farah v Malisse/Skupski - 11:30 Set 01 Winner	109605790	Cabal/Farah
Doubles Matches / Cabal/Farah v Malisse/Skupski - 11:30 Set 02 Winner	109605791	Malisse/Skupski
First Round Matches / Pospisil v Zeballos - 11:30 Set 01 Winner	109581237	Horacio Zeballos
First Round Matches / Pospisil v Zeballos - 11:30 Set 01 - Correct Score	109581240	Zeballos 7 - 6
First Round Matches / Pospisil v Zeballos - 11:30 Tie Break Played?	109581241	Yes

Figure 6.22: Tennis RSS feed results from Betfair

6.5 SPORTSBET Database

The SPORTSBET database is a comprehensive source of betting exchange market data for the supported sporting events of SPORTSBET. The historic betting exchange market data can be used to perform research and analysis, to test out potential betting systems and strategies, and to build predictive models or ratings to assess comparative chances within a sporting event. SPORTSBET database features allow bettors to analyse and produce models for sporting events that are tested on up to the second data.

Database System and Structure

The SPORTSBET database service is designed for use with the MySQL database management system (DBMS). MySQL is one of the most robust and popular databases in the world, that runs on Windows, Mac and Linux machines, and is available for free download at `www.mysql.com`. Installing MySQL is a prerequisite for using the SPORTSBET database, having installed the MySQL service, SPORTSBET can automatically build all the tables needed for SPORTSBET database services. The structure of the database is simple, with four tables in total (see Figure 6.23).

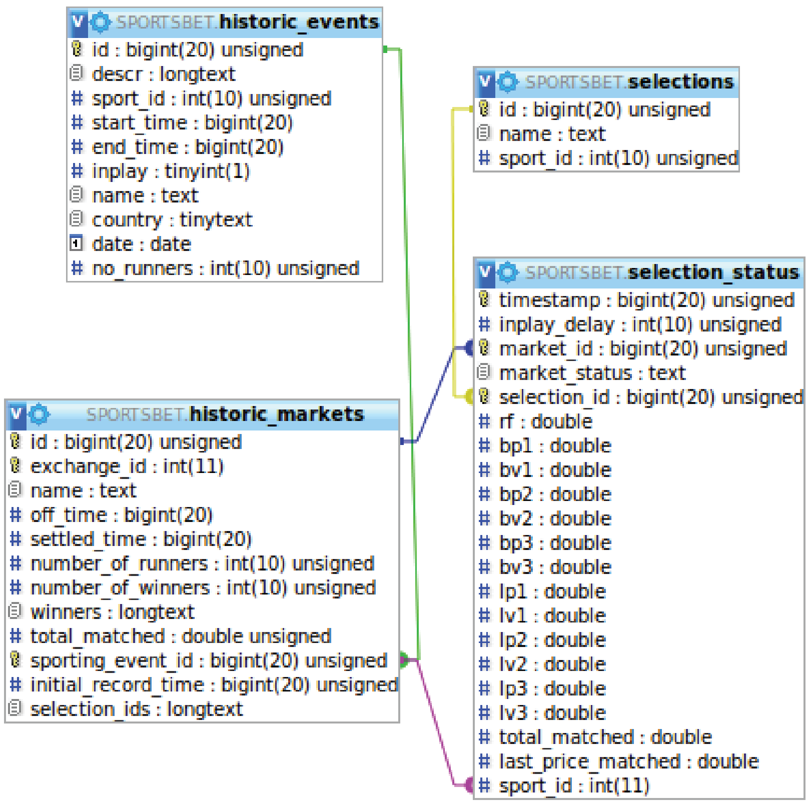


Figure 6.23: SPORTSBET's database structure

- **historic_events**: Stores the general information of a sporting event such as the date of the event, the description of the event, the start and end time of the event, the number of runners, the country the event happening and if the sporting event has markets going in-play.

- **historic_markets**: Stores information for a market, such as the name of the market, the time the market turned in-play and settled, the total amount of money matched, the number of the winners, the winners of the market and the number of selections the market has.

- **selections**: Stores the selections names and ids in each sport.
- **selection_status**: Stores the betting exchange market data.

These four tables that come as a standard feature in the database, are automatically updated whenever the user wishes to store the data of a monitoring market.

Chapter 7

Strategy Examples and Case Studies

In this chapter, we present examples of betting exchange strategies written in UBEL and finally we provide some strategy results using the back-testing, live execution and optimisation process.

7.1 Tennis Strategies

7.1.1 Simple Trading

Assume we want to try a simple trading strategy for tennis matches where in-play betting is available. The strategy is: *"If at the beginning of the match the odds to back the favourite in the market 'Match Odds' are less than 1.9, then when the match is 2 minutes in-play, back the favourite at 1.93 for £20. If the bet we placed is not matched after 20 minutes cancel it. Otherwise, when the bet is matched, immediately lay the same selection at 1.4 for £30"*. If both bets are matched and the favourite wins, the bettor will win the initial back bet of £20 at odds of 1.93 = £18.60, minus the losing lay bet of £30 at 1.4 = £12. So the total profit will be £6.60. If the underdog goes on to win, the bettor will win the lay bet of £30, minus the losing back bet of £20. So the total profit will be £10. In case the lay bet is not matched and the underdog goes on to win, then the bettor will lose the volume matched from his initial £20 back bet.

The strategy is written in UBEL as a function that takes as input a sporting event (tennis match). Using UBELs specific constructs we first define an alarm when the strategy should "wake up", which in our case is at the scheduled start time of the match. When that time comes we simply check who is the favourite of this match for the market "Match Odds". If the odds of the favourite is less than 1.9 we create a back bet for the favourite at 1.93 for £20 and then we schedule to be placed after 2 minutes. When the bet is placed we put two more alarms. The first alarm checks if after 20 minutes the placed bet has beeen matced or not. If not then we simply cancel it. The second alarm waiting for the signal that the bet was matched. If yes then we create a Lay bet for the favourite at 1.4 for £30 and we place it.

```
function tennisTrade(var sportingEvent){


  at(sportingEvent.clock == sportingEvent.startTime){


    var favourite = sportingEvent.market("Match Odds").favourite;


    if(sportingEvent.market("Match Odds").selection(favourite).bp[0] < 1.9){
        var myBet = Back.new("Match Odds", favourite, 1.93, 20, "Betfair");
        sportingEvent.placeBet([myBet], 2min);


        at(sportingEvent.clock == sportingEvent.startTime + 22min){
          if(!myBet[0].matched)
            myBet[0].cancel;
        };


        at(myBet[0].betMatched?){
          var myBet2 = Lay.new("Match Odds", favourite, 1.4, 30, "Betfair");
          sportingEvent.placeBet([myBet2]);
        };
    };
};
```

Figure 7.1 shows the visualisation of the strategy for a tennis match between Andy Murray and Gael Monfils. In this example the £20 and £30 bets were fully matched so the boxes are filled

100%. The vertical lines represent the in-play events and particularly the end of the first and second set and who was the winner of the set.

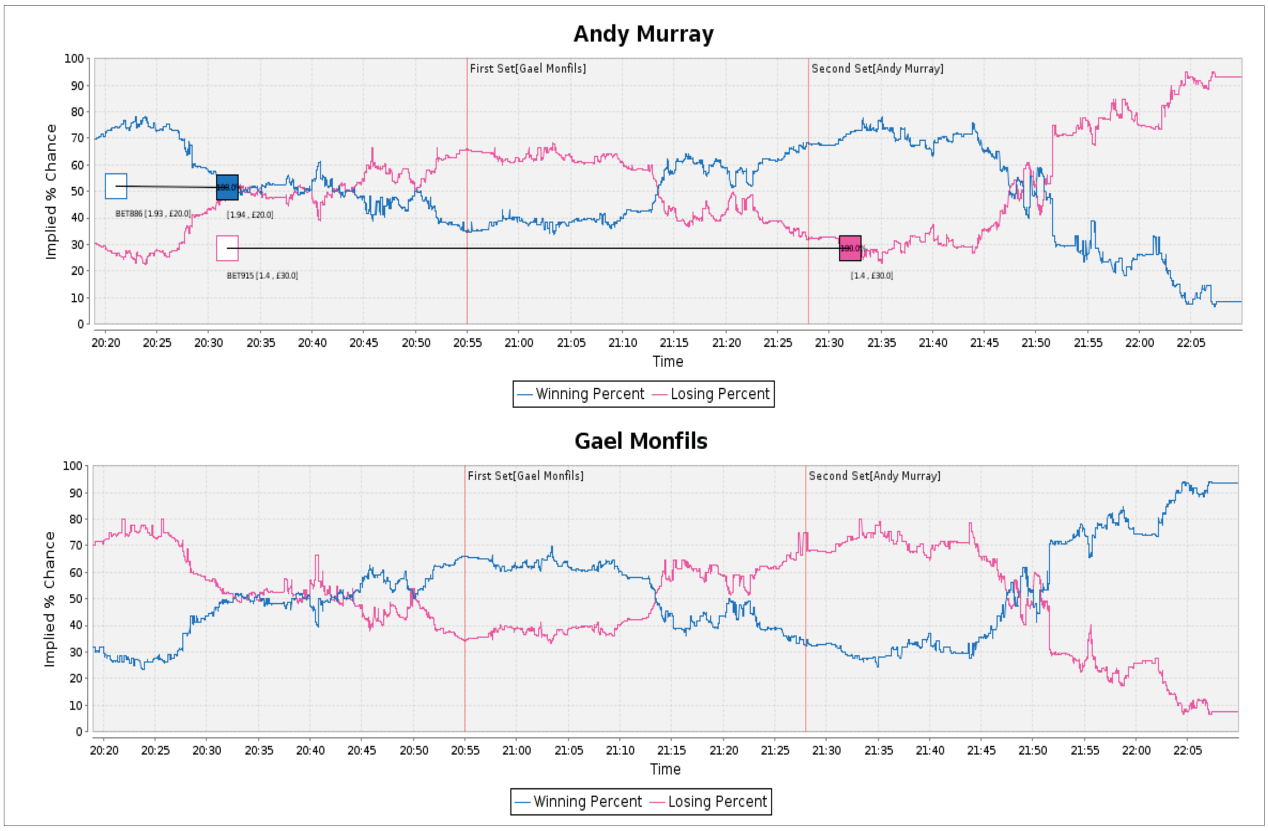


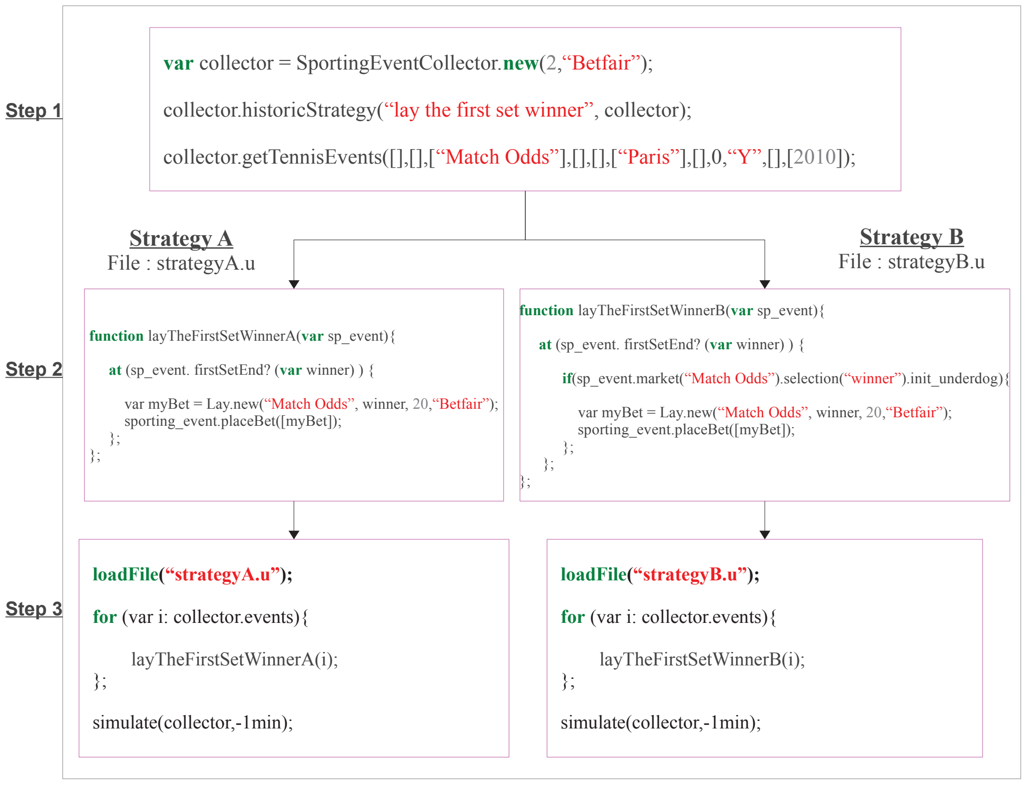
Figure 7.1: Visualisation of strategy and odds evolution in the Match Odds market over the course of a single match (Murray v Monfils).

7.1.2 Lay the First Set Winner

In this section, we describe the implementation of two betting exchange strategies for tennis matches and we evaluate them against historical data from Betfair's exchange platform. We adopt two versions of a classic strategy in tennis called "Lay the Winner of the First Set". ***Strategy A*** *is "Lay the winner of the first set for £20 at the best available odds"* while ***Strategy B*** *is "Lay the winner of the first set for £20 at the best available odds if at the beginning of the match the winner of the set was the underdog".*

Both strategies were tested at the BNP Paribas Masters 2010 which is part of the ATP World Tour Masters series. In the first step we collected the sporting events where the strategies will be tested. To do that we used the `SportingEventCollector` class. In the second step we defined the strategies using UBEL. Finally in the last step we loaded the strategies inside the complex event processing system and started the simulation. Figure 7.2 shows these steps while

Figure 7.3 shows the strategy and market visualisation in a single tennis match for Strategy A.



```
var collector = SportingEventCollector.new(2,"Betfair");

collector.historicStrategy("lay the first set winner", collector);

collector.getTennisEvents([],[],["Match Odds"],[],[],["Paris"],[],0,"Y",[],[2010]);
```

```
function layTheFirstSetWinnerA(var sp_event){
    at (sp_event. firstSetEnd? (var winner) ) {
        var myBet = Lay.new("Match Odds", winner, 20,"Betfair");
        sporting_event.placeBet([myBet]);
    };
};
```

```
function layTheFirstSetWinnerB(var sp_event){
    at (sp_event. firstSetEnd? (var winner) ) {
        if(sp_event.market("Match Odds").selection("winner").init_underdog){
            var myBet = Lay.new("Match Odds", winner, 20,"Betfair");
            sporting_event.placeBet([myBet]);
        };
    };
};
```

```
loadFile("strategyA.u");
for (var i: collector.events){
    layTheFirstSetWinnerA(i);
};
simulate(collector,-1min);
```

```
loadFile("strategyB.u");
for (var i: collector.events){
    layTheFirstSetWinnerB(i);
};
simulate(collector,-1min);
```

Figure 7.2: Back-testing steps

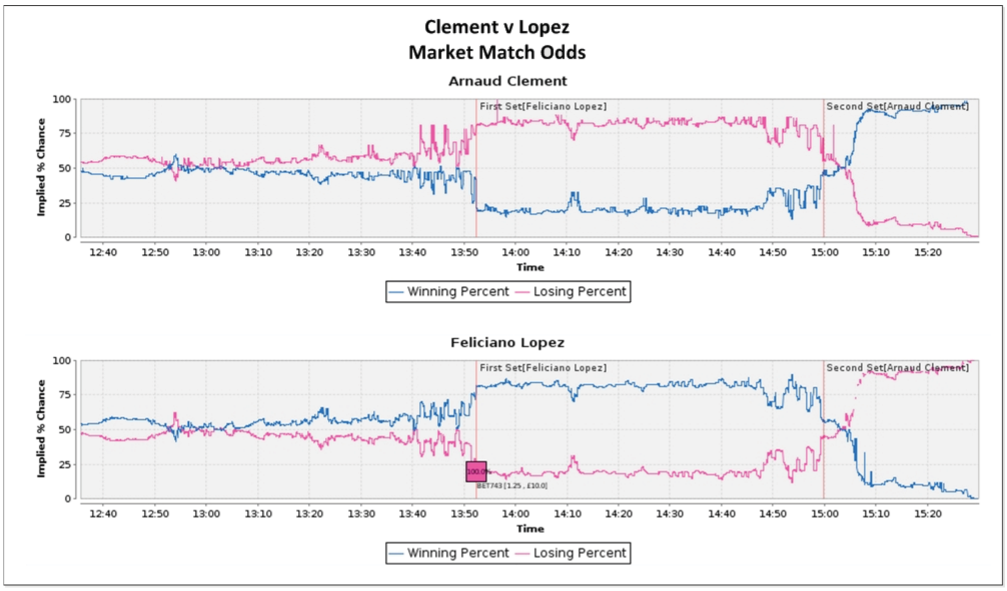
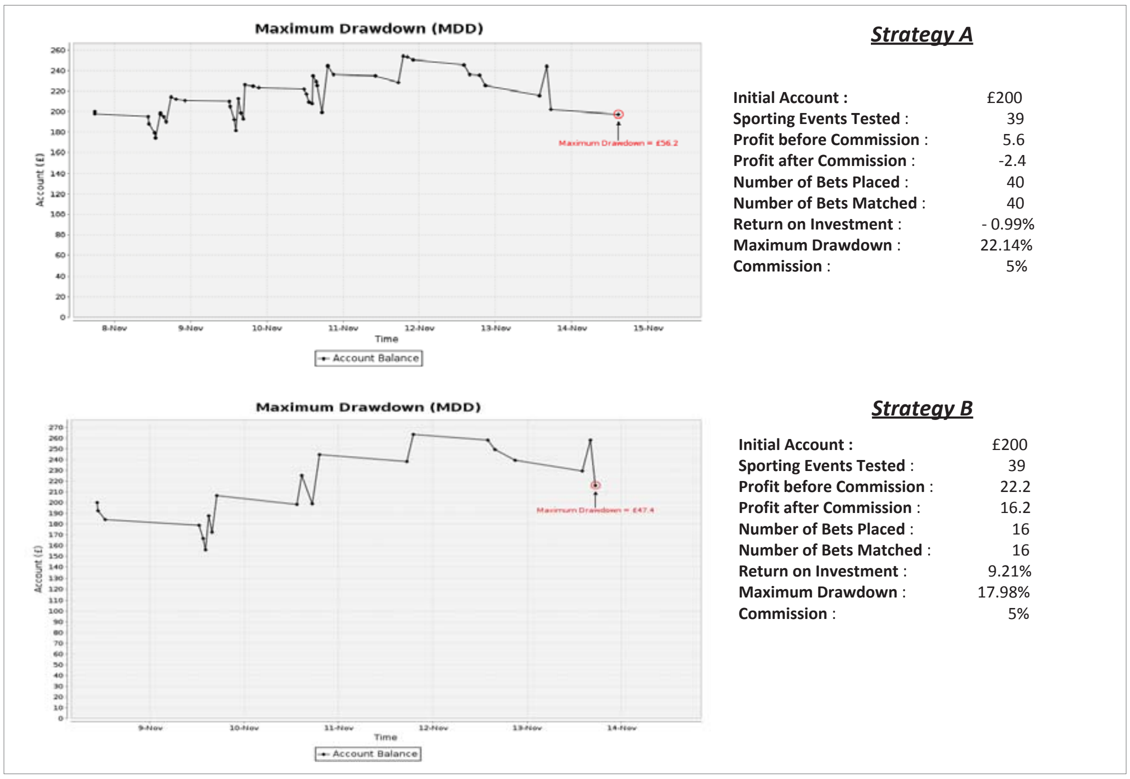


Figure 7.3: Visualisation of strategy and odds evolution in the Match Odds market over the course of a single match (Clement v. Lopez).

As explained in Section 6.3 in order to detect that a set has been won (as shown in Figure 7.3) we analysed the price changes in the market "Set Betting". Specifically, the conclusion of a set usually results in the odds of the selections in the “Set Betting” market that are now no longer possible reaching 1000.

Figure 7.4 shows the results from the back-testing.



Strategy A

Initial Account :	£200
Sporting Events Tested :	39
Profit before Commission :	5.6
Profit after Commission :	-2.4
Number of Bets Placed :	40
Number of Bets Matched :	40
Return on Investment :	- 0.99%
Maximum Drawdown :	22.14%
Commission :	5%

Strategy B

Initial Account :	£200
Sporting Events Tested :	39
Profit before Commission :	22.2
Profit after Commission :	16.2
Number of Bets Placed :	16
Number of Bets Matched :	16
Return on Investment :	9.21%
Maximum Drawdown :	17.98%
Commission :	5%

Figure 7.4: Strategy results over 39 matches in the 2010 BNP Paribas Masters ATP tennis tournament.

As we can see **Strategy A** placed 40 bets on 39 tennis matches. The number of bets placed and matched is bigger than the number of tennis matches because in one of the tennis matches the available volume to lay the winner at the best odds was less than the volume of the bet strategy placed. So an extra bet was placed at the second best price to match the remainder of the initial bet. **Strategy A** makes a small profit of £5.60, but due to Betfair’s default commission of 5%, the final result is a loss of £2.40. Moreover we can see that the Return on Investment is -0.99% and that the maximum amount of money we risked was £56.20. **Strategy B** placed 16 bets on 39 tennis matches. **Strategy B** makes a small profit of £22.20 but due to the default commission, the final result is a profit of £16.20. The Return on Investment is 9.21% and the maximum amount of money risked was £47.40.

7.1.3 High Frequency Trading Strategy

In this section we present a high frequency trading strategy. The strategy is: *"If the odds of the favourite are more than 1.07 and less than 1.7 then lay the favourite at the best back odds for £5 and at the same time back the favourite at the best lay odds for £5. When both bets are matched repeat the process"*.

Betfair acceptable odds in UBEL

```
function highFreq(var sportingEvent){


  var strategyTag = Tag.new;
  var reset = false;
  var matched = 0;
  var matchOdds = sportingEvent.market("Match Odds");
  strategyTag:{
    at(matchOdds.marketUpdated?){
      if(!reset){
        reset = true;
        if(matchOdds.selection("favourite").bp[0]<1.7 && matchOdds.selection("favourite").bp[0]>1.07){
            var bp = matchOdds.selection("favourite").bp[0]&
            var lp = matchOdds.selection("favourite").lp[0];
            var myBackBet = Back.new("Match Odds","favourite",lp,5,"Betfair");
            var myLayBet = Lay.new("Match Odds","favourite",bp,5,"Betfair");
            var bets = sportingEvent.placeBet([myBackBet,myLayBet]);
            at(bets[0].matched)
              matched = matched+1;
            at(bets[1].matched)
              matched = matched+1;
            at(matched==2){
              matched = 0;
              reset = false;
            };
        }
        else{
          reset = false;
        };
      };
    };
  };
};
```

The strategy was tested on 34 tennis matches of the Roland-Garros 2013 tournament. Figure

7.5 shows a visualisation of the trading strategy and Table 7.1 summarises the results.

Table 7.1: High frequency trading strategy results in tennis

Profit (£)	-20.10
Profit after commission (£)	-20.61
MDD (£)	100.45
MDD (%)	50.05%
MRU (£)	60.09
MRU (%)	59.94%
ROI (%)	-13.93%
PROM (%)	-0.86%
RRR (%)	-20.52%
RAR (%)	-0.94%
Perfect Profit (£)	130.96
Total Winnings (£)	9.64
Total Loses (£)	-30.25
#Number of wins	7
#Number of loses	18
#Maximum number of consecutive wins	6
#Maximum number of consecutive loses	8
Strategy Efficiency	-15.74%

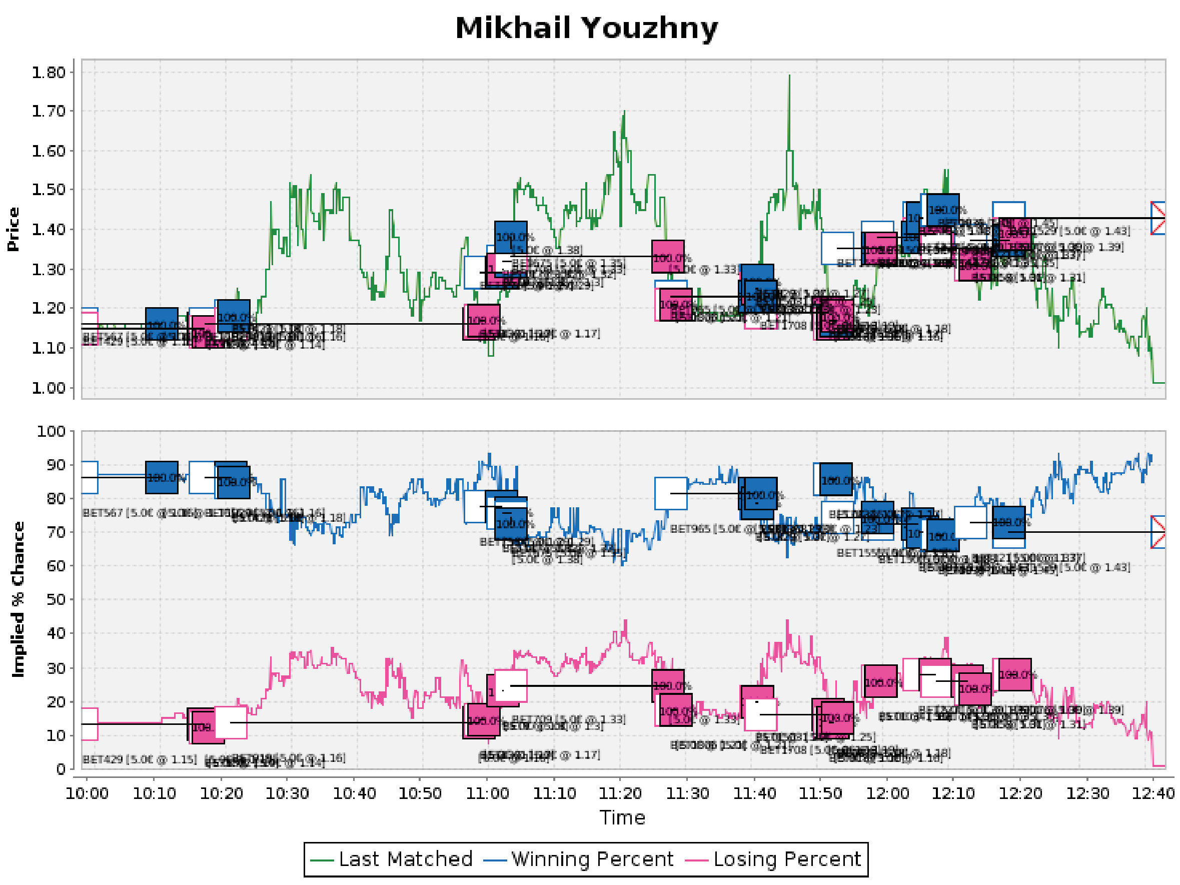


Figure 7.5: Visualisation of a high frequency trading strategy and odds evolution in the Match Odds market over the course of a single match (Youzhny v. Delbonis).

Let us remove the last step of the above strategy, then the strategy will be : *"If the odds of the favourite are more than 1.07 and less than 1.7 then lay the favourite at the best back odds for £5 and at the same time back the favourite at the best lay odds for £5."*.

Betfair acceptable odds in UBEL

```
function trader(var sportingEvent){
  var strategyTag = Tag.new;
  var reset = false;
  var matchOdds = sportingEvent.market("Match Odds");
  strategyTag:{
    at(matchOdds.marketUpdated?){
      if(!reset){
        reset = true;
          if(matchOdds.selection("favourite").bp[0]<1.7 && matchOdds.selection("favourite").bp[0]>1.07){
            var bp = matchOdds.selection("favourite").bp[0];
            var lp = matchOdds.selection("favourite").lp[0];
            var myBackBet = Back.new("Match Odds","favourite",lp,5,"Betfair");
            var myLayBet = Lay.new("Match Odds","favourite",bp,5,"Betfair");
            var bets = sportingEvent.placeBet([myBackBet,myLayBet]);
            strategyTag.stop;
          }
          else{
            reset = false;
};};};};};
```

Figure 7.6 shows a visualisation of the trading strategy and Table 7.2 summarises the results.

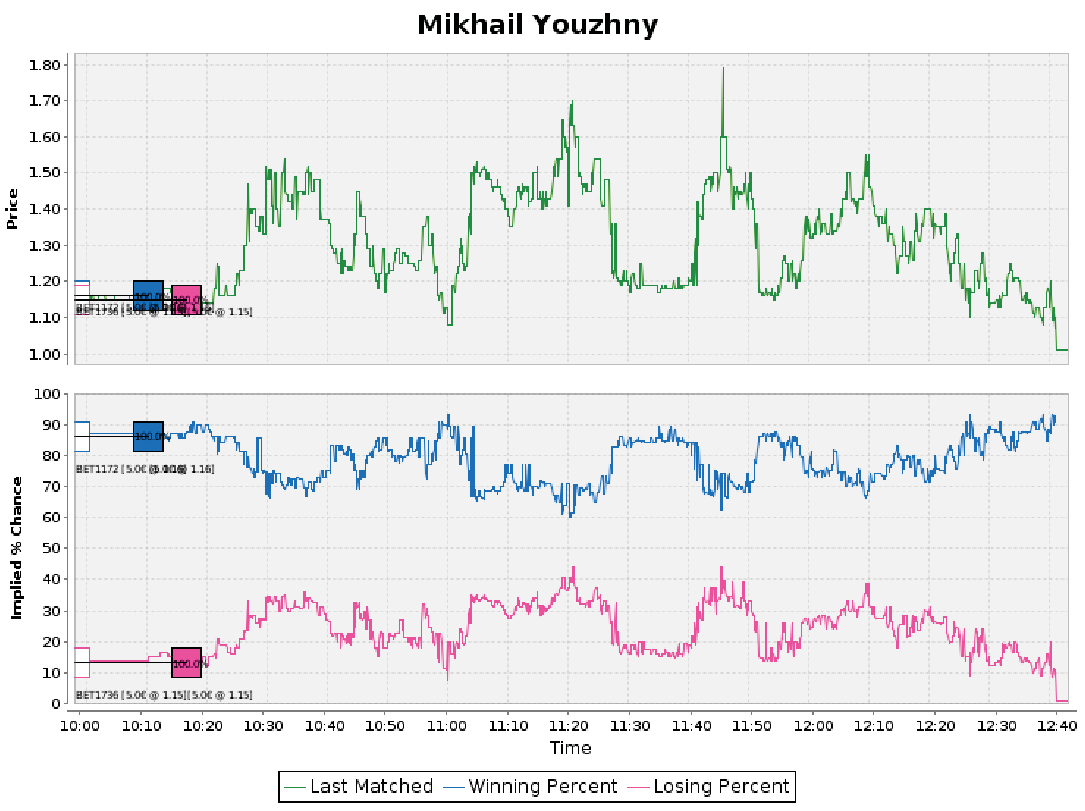


Figure 7.6: Visualisation of a trading strategy and odds evolution in the Match Odds market over the course of a single match (Youzhny v. Delbonis).

Table 7.2: Trading strategy results in tennis

Profit (£)	0.65
Profit after commission (£)	0.54
MDD (£)	44.71
MDD (%)	22.25%
MRU (£)	44.31
MRU (%)	28.36%
ROI (%)	0.39%
PROM (%)	0.08%
RRR (%)	1.21%
RAR (%)	0.03%
Perfect Profit (£)	9.83
Total Winnings (£)	2.04
Total Loses (£)	-1.50
#Number of wins	18
#Number of loses	1
#Maximum number of consecutive wins	16
#Maximum number of consecutive loses	1
Strategy Efficiency	5.52%

7.2 Horse Racing Strategies

7.2.1 Live Algorithmic Trading

In this section we present a live algorithmic trading strategy example for horse racing. The strategy is: *"30 seconds before a horse race start lay the top K favourites of the horse race, at the best available odds for £5, only if the liability is less than £50 and the number of runners is greater than 6"*. The input of the strategy is the sporting event, the number of horses you want to lay (K), the maximum allowed liability and the stake.

The strategy was tested for $K = 1, K = 2, K = 3$ and $K = 4$ in 177 UK horse races in 2012. Figure 7.8 shows the strategy performance in each case and table 7.3 summarises the results.

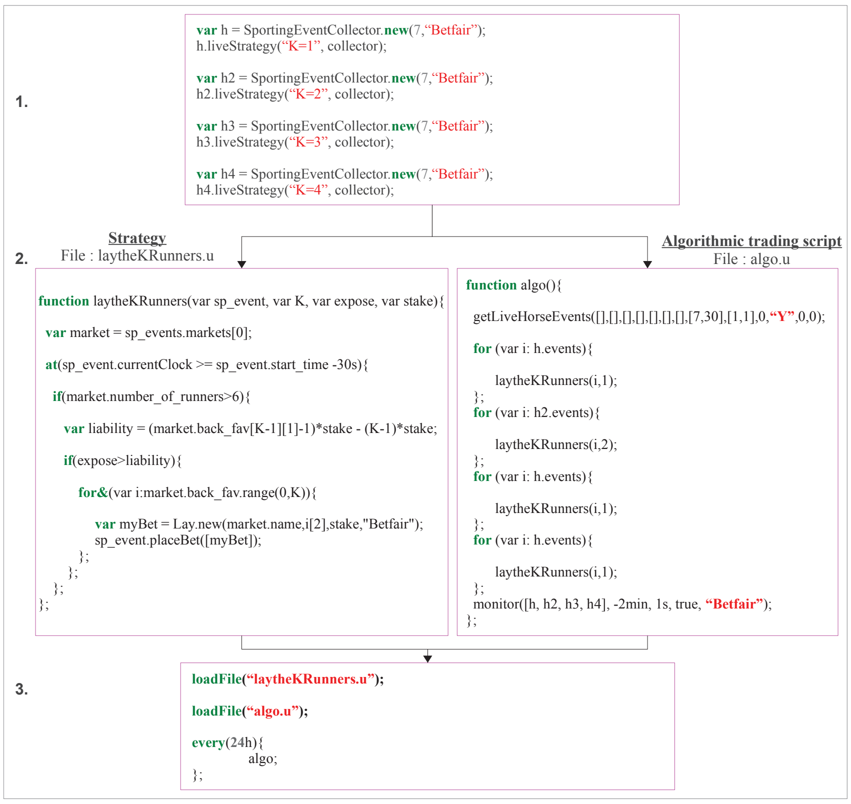
strategy function

```
function laytheKRunners(var sportingEvent, var K, var expose, var stake){


  var market = sportingEvent.markets[0];

  at(sportingEvent.clock >= sportingEvent.startTime -30s){


    if(market.numberOfRunners>6){

        var liability = (market.backFav[K-1][1]-1)*stake - (K-1)*stake;

        if(expose>liability){


          for&(var i:market.backFav.range(0,K)){


            var myBet = Lay.new(market.name,i[2],stake,"Betfair");

            sportingEvent.placeBet([myBet]);

          };

        };

    };

};
```

In order to execute the strategy daily on the available horse races, an extra function has to be created which is going to be executed every 24 hours. This function will be responsible to collect all the available daily horse races that matches user's criteria, apply the strategy on those markets and finally request to monitor them.

algorithmic trading function

```
function algo(){

  getLiveHorseEvents([],[],[],[],[],[],[],[7,30],[1,1],0,"Y",0,0);

  for (var i: h.events){

        laytheKRunners(i,1);

  };

  for (var i: h2.events){

        laytheKRunners(i,2);

  };

  for (var i: h.events){

        laytheKRunners(i,1);

  };

  for (var i: h.events){

        laytheKRunners(i,1);

  };

  monitor([h, h2, h3, h4], -2min, 1s, true, "Betfair");

};
```



```
var h = SportingEventCollector.new(7,“Betfair”);
h.liveStrategy(“K=1”, collector);

var h2 = SportingEventCollector.new(7,“Betfair”);
h2.liveStrategy(“K=2”, collector);

var h3 = SportingEventCollector.new(7,“Betfair”);
h3.liveStrategy(“K=3”, collector);

var h4 = SportingEventCollector.new(7,“Betfair”);
h4.liveStrategy(“K=4”, collector);
```

```
function laytheKRunners(var sp_event, var K, var expose, var stake){

 var market = sp_events.markets[0];

 at(sp_event.currentClock >= sp_event.start_time -30s){

  if(market.number_of_runners>6){

    var liability = (market.back_fav[K-1][1]-1)*stake - (K-1)*stake;

    if(expose>liability){

      for&(var i:market.back_fav.range(0,K)){

         var myBet = Lay.new(market.name,i[2],stake,"Betfair");
         sp_event.placeBet([myBet]);
      };
    };
  };
};
```

```
function algo(){

 getLiveHorseEvents([],[],[],[],[],[],[],[7,30],[1,1],0,“Y”,0,0);

 for (var i: h.events){

     laytheKRunners(i,1);
 };
 for (var i: h2.events){

     laytheKRunners(i,2);
 };
 for (var i: h.events){

     laytheKRunners(i,1);
 };
 for (var i: h.events){

     laytheKRunners(i,1);
 };
 monitor([h, h2, h3, h4], -2min, 1s, true, “Betfair”);
};
```

```
loadFile(“laytheKRunners.u”);

loadFile(“algo.u”);

every(24h){
        algo;
};
```

Figure 7.7: Horse racing live algorithmic trading steps

Table 7.3: Horse racing algorithmic trading results

K	Profit (£)	Profit after commission (£)	Total Account (£)	MDD (£)
1	-152.94	-181.44	-157.87	258.84
2	-53.91	-96.85	-73.27	238.39
3	7.69	-42.02	-19.03	230.62
4	174.34	121.61	145.18	259.97

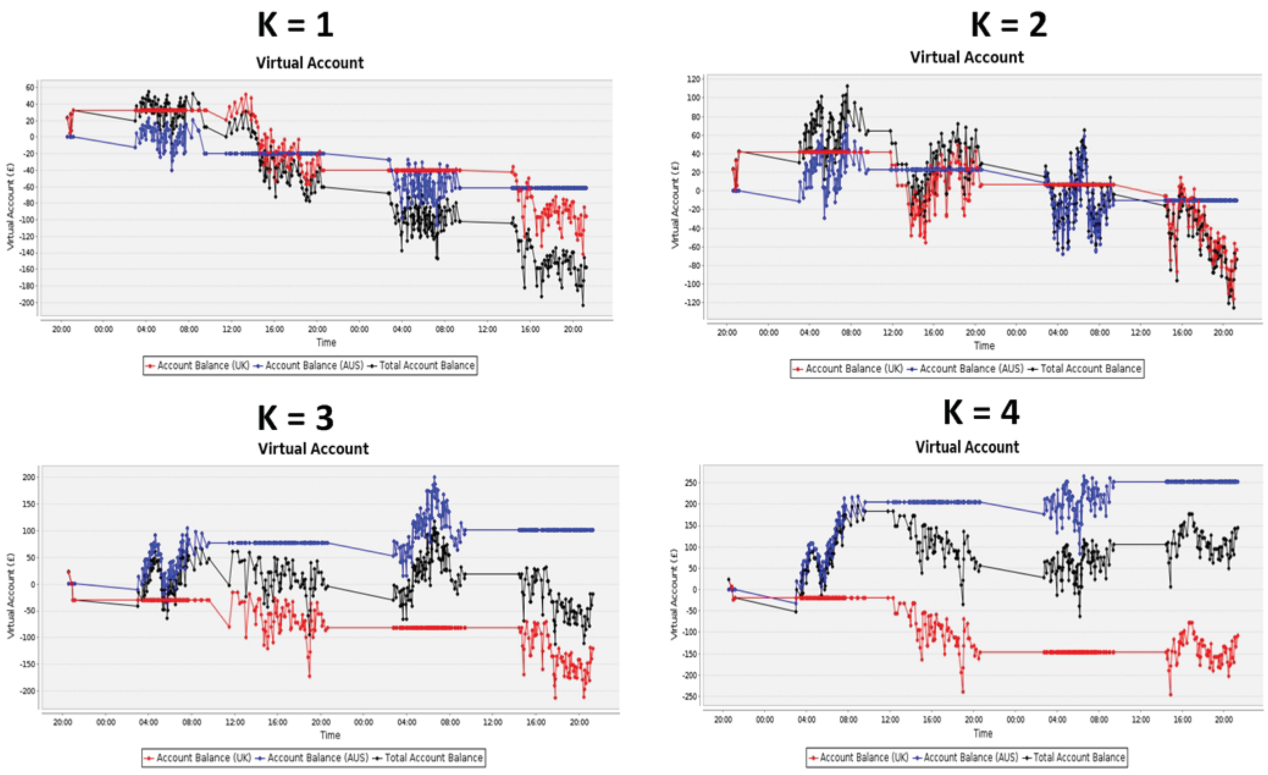


Figure 7.8: Horse racing algorithmic trading results

7.2.2 Simple Trading

The strategy is: *"30 seconds before a horse race start, back the favourite of the horse race, at the best available odds for £30. When the back bet is matched, lay the same selection of the matched back bet so in case the lay bet matched you win 20 % from your initial stake. When both bets are matched repeat the process"*. The input of the strategy is the sporting event, the stake and the percentage you want to win from the initial stake.

```
function horseTrade(var horseRace,var stake,var percent){

  var cycle = Event.new;
  var market  = horseRace.markets[0];
  var ex = "Betfair";

  at(horseRace.clock >= horseRace.startTime - 30s){
    market.keepBets = true;
    cycle!;
  };

  at(cycle?){
    var backBet = horseRace.placeBet([Back.new(market.name,market.favourite,stake,ex)]);
    var betTag = Tag.new;

    betTag: {

      at(backBet[0].matched){
        var betOdds = greenOdds("lay",percent,backBet[0].odds,stake,ex);
        var layStake = greenStake(backBet[0].odds,stake,betOdds,ex);
        var name = backBet[0].selectionName;
        var layBet = horseRace.placeBet([Lay.new(market.name,name,betOdds,layStake,ex)]);

        at(layBet[0].matched){
          if(market.status=="ACTIVE"){
            betTag.stop & cycle!;
          };
        };
      };
    };
  };
};
```

The strategy was tested in 331 UK horse races in 2013. Table 7.4 summarises the results and Figure 7.9 shows the strategy and market visualisation in a single horse racing market.

Table 7.4: Horse racing algorithmic trading results

Profit (£)	-742.28
Profit after commission (£)	-765.76
MDD (£)	894.24
MDD (%)	444.04%
MRU (£)	177.73
MRU (%)	132.97%
ROI (%)	-3.51%
PROM (%)	-30.56%
RRR (%)	-85.63%
RAR (%)	-20.21%
Perfect Profit (£)	2875.25
Total Winnings (£)	446.03
Total Loses (£)	-1211.79
#Number of wins	269
#Number of loses	62
#Maximum number of consecutive wins	15
#Maximum number of consecutive loses	3
Strategy Efficiency	-26.63%

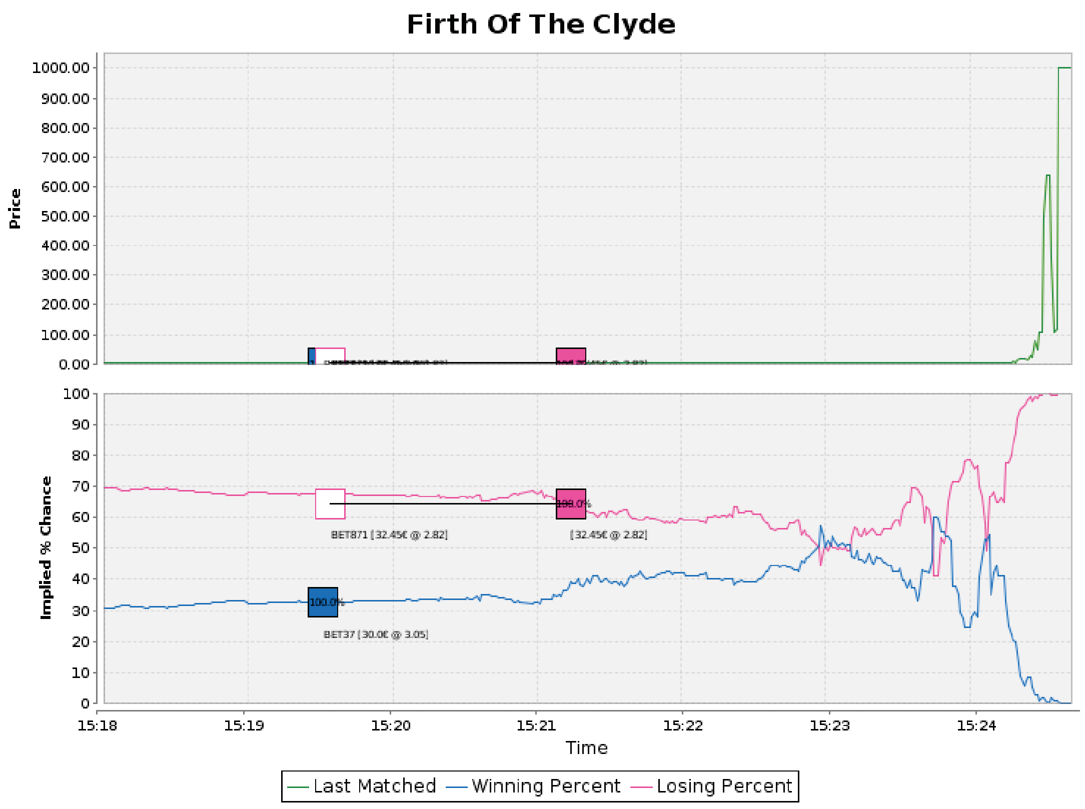


Figure 7.9: Visualisation of horse racing strategy and odds evolution over the course of a single selection (Firth Of The Clyde).

7.2.3 Walk-Forward Analysis

A horse racing case study was used as a means of demonstrating and evaluating the results yielded by the SPORTSBET optimisation platform. The trading strategy is: *"1 minute before a race start, if the odds on the favourite are more than X_1 where $X_1 \in \mathbb{R}$, then back a range of horses $X_2,\ X_3$ where $X_2 \leq X_3,\ X_2 > 0,\ X_3 \leq 5,\ X_2,\ X_3 \in \mathbb{Z}$, with a total of £40 such that the profit is spread evenly across all the backed horses"*.

So we want to find the best combination of $X_1,\ X_2,\ X_3$. The strategy was tested in 990 markets using 10 time windows. The chosen objective function was the PROM and each time window consisted of 99 markets. The step size of the Walk-Forward Analysis was three time windows. Table 7.5 shows the Walk-Forward Efficiency of the strategy in each time window. Table 7.6 shows an optimisation sample of the time windows 4, 5 and 6, where TW is the current Time Window, I is the iteration of the searching algorithm, *Eval.* the number of evaluation (how many times a back-test performed), $Fitness$ is the fitness function, $ProfitAfterC.$ is the profit after commission, PP is the perfect profit, C_W is the maximum number of consecutive wins and C_L is the maximum number of consecutive losses, W the total number of wins and L the total number of loses.

Table 7.5: Walk-Forward Efficiency

Time Window	X	Profit after commission	MDD	Winning Perc. (%)	WFE (%)
4	[3.95, 1, 5]	-116.51	236.11	50	-196.39
5	[5.4, 4, 5]	758	80	66.66	128.01
6	[3.4, 3, 5]	-33.85	373.43	22.58	-8.18
7	[3.5, 1, 3]	30.35	200	63.33	13.7
8	[2.6, 1, 4]	442.38	239.6	44	33.3
9	[2.8, 1, 3]	123.29	265.39	39.5	19.4
10	[3.1, 1, 4]	218.43	178.6	42.43	35.36

So we have $\overline{WFE} = 3.6$, which means the strategy did not pass the Walk-Forward Analysis test. Research has demonstrated that robust trading strategies have WFEs greater than 50-60 percent [93]. In that case, in order to choose which set of parameters will be used for the real time trading, we use the metric Sharpe ratio that was described in Section 5.4.

Table 7.6: Optimisation process of Time Windows 4 - 5 - 6 for a dutch horse racing trading strategy

TW	*I*	*Eval.*	*X*	*Fitness*	*Profit After C.*	*MDD*	*ROI*	*Profit*	*MDD%*	*MRU*	*MRU%*	*PP*	*PROM*	*RAR*	*RRR*	*SE*	C_W	*CL*	*W*	*L*
6	4	34	[2.76, 1, 4]	-9.87	-144.81	657.37	-2.74	-0.85	179.53	411.61	1005.62	6036.9	-9.87	-4.37	-22.03	-2.4	4	4	60	72
6	4	35	[3.65, 3, 4]	15.42	1032.00	831.2	33.51	1222.4	55.35	1541.6	3954	21720.8	15.42	28.18	124.16	4.75	2	14	12	65
6	4	36	[3.25, 2, 4]	15.19	586.74	607.48	14.67	764.4	62.43	853.09	710.91	10811.5	15.19	18.25	96.59	5.43	3	8	30	70
6	4	37	[3.35, 2, 3]	-36.71	-302.18	1280.98	-8.21	-153.38	177.72	698	224.6	21433.2	-36.71	-6.62	-23.59	-1.41	2	16	13	79
6	4	38	[3.15, 3, 4]	8.33	963.00	1052.8	22.93	1202.4	59.34	1814.2	4635.5	29477.4	8.33	23.46	91.47	3.27	2	19	15	90
6	4	39	[3.5, 1, 5]	20.55	423.17	254.43	12.75	512.81	50.93	531.57	580.27	2811.2	20.55	16.87	166.32	15.05	5	3	51	32
6	4	40	[2.58, 3, 5]	-13.51	69.81	657.52	1.25	302.96	82.91	888.59	1030.13	21087.9	-13.51	2.11	10.62	0.33	3	11	31	109
6	5	41	[3.8, 1, 5]	9.08	184.15	259.15	6.67	256.99	81.54	325.49	554.85	2430.74	9.08	7.31	71.06	7.58	5	3	39	30
6	5	42	[3.6, 2, 5]	18.02	503.20	356.66	16.13	626.28	47.13	716.66	597.22	5482.03	18.02	18.55	141.09	9.18	4	5	32	46
6	5	43	[3.65, 1, 4]	13.52	326.75	362.21	10.61	426.05	61.17	472.13	393.48	3924.68	13.52	11.99	90.21	8.33	3	4	38	39
6	5	44	[3.75, 3, 4]	4.36	768.00	831.2	25.6	942.4	67.16	1237.6	3687	21082.4	4.36	20.97	92.4	3.64	2	14	11	64
6	5	45	[3.8, 2, 4]	-0.06	225.62	503.98	8.17	342.17	77.51	650.68	134984	7827.47	-0.06	7.5	44.77	2.88	3	7	19	50
6	5	46	[3.4, 2, 5]	33.05	827.28	302.86	23.5	973.73	34.02	1040.74	867.28	6116.46	33.05	31.75	273.16	13.53	4	5	39	49
6	5	47	[3.35, 2, 4]	11.11	494.60	739.42	13.44	656.89	73	892.9	744.08	10127.5	11.11	14.22	66.89	4.88	3	8	27	65
6	5	48	[3.8, 2, 5]	8.9	299.48	331.47	10.85	403.42	62.79	552.94	691.18	4887.75	8.9	11.25	90.35	6.13	4	5	27	42
6	5	49	[3.25, 1, 4]	-0.7	12.38	583.64	0.31	126.72	123.85	368.21	357.41	4790.36	-0.7	0.39	2.12	0.26	3	6	46	54
6	5	50	[3.1, 2, 5]	23.91	634.27	359.98	14.55	802.14	54.3	938.15	1348.28	7515.43	23.91	23.32	176.2	8.44	4	8	45	64
6	6	51	[3.65, 2, 5]	19.87	543.20	316.66	17.64	666.28	41.84	756.66	630.55	5440.85	19.87	20.63	171.54	9.98	4	4	32	45
6	6	52	[3.55, 2, 5]	26.75	691.44	305.36	21.34	824.43	40.35	904.9	754.08	5670.26	26.75	26.48	226.44	12.19	4	5	35	46
6	6	53	[3.25, 2, 4]	15.19	586.74	607.48	14.67	764.4	62.43	853.09	710.91	10811.5	15.19	18.25	96.59	5.43	3	8	30	70
6	6	54	[2.84, 2, 5]	15.78	542.75	479.98	10.77	731.06	77.69	760.59	1093.11	8682.74	15.78	18.34	113.08	6.25	4	10	50	76
6	6	55	[3.4, 3, 5]	41.36	1242.24	440	35.29	1438.15	28.77	1673.54	4283.84	12402.4	41.36	43.13	282.33	10.02	3	10	26	62
6	6	56	[3.45, 2, 5]	31.36	792.28	302.86	23.58	932.67	35.42	1005.73	838.11	5884.54	31.36	30.41	261.6	13.46	4	5	37	47
6	6	57	[3.25, 2, 4]	15.19	586.74	607.48	14.67	764.4	62.43	853.09	710.91	10811.5	15.19	18.25	96.59	5.43	3	8	30	70
6	6	58	[3.75, 2, 5]	18.4	510.30	316.66	17.01	629.54	43.74	723.76	603.13	5268.96	18.4	19.38	161.15	9.69	4	5	31	44
6	6	59	[3.4, 2, 5]	33.05	827.28	302.86	23.5	973.73	34.02	1040.74	867.28	6116.46	33.05	31.75	273.16	13.53	4	5	39	49
6	6	60	[3.6, 2, 5]	18.02	503.20	356.66	16.13	626.28	47.13	716.66	597.22	5482.03	18.02	18.55	141.09	9.18	4	5	32	46
6	7	61	[3.55, 3, 4]	7.91	872.00	911.2	26.91	1062.4	60.68	1541.6	3954	22841.8	7.91	22.81	95.7	3.82	2	15	12	69
6	7	62	[3.85, 2, 5]	8.9	299.48	331.47	10.85	403.42	62.79	552.94	691.18	4887.75	8.9	11.25	90.35	6.13	4	5	27	42
6	7	63	[3.6, 2, 4]	11.78	501.33	623.98	16.07	642.92	59.66	925.91	771.59	8717.19	11.78	15.44	80.34	5.75	3	8	23	55
6	7	64	[3.55, 2, 5]	26.75	691.44	305.36	21.34	824.43	40.35	904.9	754.08	5670.26	26.75	26.48	226.44	12.19	4	5	35	46
6	7	65	[3.4, 3, 5]	41.36	1242.24	440	35.29	1438.15	28.77	1673.54	4283.84	12402.4	41.36	43.13	282.33	10.02	3	10	26	62
6	7	66	[3.6, 2, 4]	11.78	501.33	623.98	16.07	642.92	59.66	925.91	771.59	8717.19	11.78	15.44	80.34	5.75	3	8	23	55
6	7	67	[3.55, 2, 5]	26.75	691.44	305.36	21.34	824.43	40.35	904.9	754.08	5670.26	26.75	26.48	226.44	12.19	4	5	35	46

7.3 Financial Strategy

In this section we present a simple trading strategy for the betfair financials FTSE100 and WallSt 20 minute markets, where you can bet on the selections “FTSE100 Down”, “FTSE100 Up or Same” and “WallSt30 Down”, “WallSt30 Up or Same” respectively. The strategy is : *“1 minute before the start of each market back both selections at odds of x where $x \in \mathbb{R}$, $2 \leq x \leq 2.5$”*.

So the question is whether the strategy is successful and if so what is the best x?

```
function financeStr(var sportingEvent, var x){


  var strategyTag = Tag.new;


  strategyTag :{
    at(sportingEvent.clock >= sportingEvent.startTime -1min){
      for&(var s:sportingEvent.markets[0].selections){
        var myBet = Back.new(sportingEvent.markets[0].name, s.name, x, 5,"Betfair");
        sportingEvent.placeBet([myBet]);
      };
      strategyTag.stop;
    };
  };
};
```

We applied the optimisation algorithm in order to see if the strategy is promising. The strategy was tested and optimised on 160 financial markets and the metric of Profit After Commission was chose as an objective function. Tables 7.7, 7.8 and 7.9 showing the optimisation results while Figure 7.10 shows the graph from the optimisation process.

From the results we can see that the best $x = 2.22$ with Profit After Commission $= -140$. So the strategy is rejected as it is not profitable.

Table 7.7: Optimisation process in financial trading strategy

TW	*I*	*Eval.*	*X*	*Fitness*	*Profit After C.*	*MDD*	*ROI*	*Profit*	*MDD%*	*MRU*	*MRU%*	*PROM*	*RAR*	*RRR*	*SE*	C_W	*CL*	*W*	*L*
1	1	1	[2.00]	-143.84	-143.84	146.51	-23.77	-140.65	73.25	19.83	13.56	-6.4	-6.27	-98.18	-56.90	2	7	18	46
1	1	2	[2.06]	-158.11	-158.11	163.11	-26.13	-154.89	81.55	20.39	14.25	-6.73	-6.8	-96.93	-56.07	6	5	47	46
1	1	3	[2.08]	-158.98	-158.98	163.98	-26.28	-155.55	81.99	20.77	14.40	-6.79	-6.83	-96.95	-53.91	6	5	46	47
1	1	4	[2.06]	-158.11	-158.11	163.11	-26.13	-154.89	81.55	20.39	14.25	-6.73	-6.8	-96.93	-56.07	6	5	47	46
1	2	5	[2.12]	-143.96	-143.96	148.96	-23.80	-139.88	74.48	21.53	14.10	-6.17	-6.26	-96.64	-46.48	6	5	44	47
1	2	6	[2.06]	-158.11	-158.11	163.11	-26.13	-154.89	81.55	20.39	14.25	-6.73	-6.8	-96.93	-56.07	6	5	47	46
1	2	7	[2.16]	-148.43	-148.43	153.43	-24.53	-144.19	76.71	21.02	14.69	-6.39	-6.43	-96.74	-45.87	6	4	42	48
1	2	8	[2.08]	-158.98	-158.98	163.98	-26.28	-155.55	81.99	20.77	14.40	-6.79	-6.83	-96.95	-53.91	6	5	46	47
1	3	9	[2.24]	-154.15	-154.15	159.15	-25.48	-150.11	79.58	18.81	11.64	-6.7	-6.65	-96.86	-45.25	6	5	36	49
1	3	10	[2.28]	-169.68	-169.68	174.68	-28.05	-166.20	87.34	16.98	13.49	-7.4	-7.22	-97.14	-49.62	6	5	32	50
1	3	11	[2.26]	-160.75	-160.75	165.75	-26.57	-156.79	82.87	19.19	11.87	-7.01	-6.89	-96.98	-46.35	6	5	34	50
1	3	12	[2.18]	-147.12	-147.12	152.13	-24.32	-142.81	76.06	21.21	14.74	-6.34	-6.38	-96.71	-44.40	6	4	41	48
1	4	13	[2.48]	-160.69	-160.69	165.69	-26.56	-156.68	82.84	21.40	30.91	-7.12	-6.89	-96.98	-39.76	6	8	25	50
1	4	14	[2.46]	-163.17	-163.17	168.17	-26.97	-159.29	84.09	20.93	30.87	-7.22	-6.98	-97.03	-41.06	6	8	25	50
1	4	15	[2.06]	-158.11	-158.11	163.11	-26.13	-154.89	81.55	20.39	14.25	-6.73	-6.8	-96.93	-56.07	6	5	47	46
1	4	16	[2.26]	-160.75	-160.75	165.75	-26.57	-156.79	82.87	19.19	11.87	-7.01	-6.89	-96.98	-46.35	6	5	34	50
1	5	17	[2.20]	-143.25	-143.25	148.25	-23.68	-138.74	74.13	21.40	14.80	-6.18	-6.24	-96.63	-42.24	6	4	41	48
1	5	18	[2.20]	-143.25	-143.25	148.25	-23.68	-138.74	74.13	21.40	14.80	-6.18	-6.24	-96.63	-42.24	6	4	41	48
1	5	19	[2.26]	-160.75	-160.75	165.75	-26.57	-156.79	82.87	19.19	11.87	-7.01	-6.89	-96.98	-46.35	6	5	34	50
1	5	20	[2.24]	-154.15	-154.15	159.15	-25.48	-150.11	79.58	18.81	11.64	-6.7	-6.65	-96.86	-45.25	6	5	36	49
1	6	21	[2.14]	-151.07	-151.07	156.07	-24.97	-146.84	78.04	21.91	15.39	-6.52	-6.53	-96.80	-46.16	6	5	42	49
1	6	22	[2.24]	-154.15	-154.15	159.15	-25.48	-150.11	79.58	18.81	11.64	-6.7	-6.65	-96.86	-45.25	6	5	36	49
1	6	23	[2.10]	-154.58	-154.58	159.58	-25.55	-150.92	79.79	21.15	14.55	-6.61	-6.67	-96.87	-51.01	6	5	46	47
1	6	24	[2.26]	-160.75	-160.75	165.75	-26.57	-156.79	82.87	19.19	11.87	-7.01	-6.89	-96.98	-46.35	6	5	34	50
1	7	25	[2.24]	-154.15	-154.15	159.15	-25.48	-150.11	79.58	18.81	11.64	-6.7	-6.65	-96.86	-45.25	6	5	36	49
1	7	26	[2.20]	-143.25	-143.25	148.25	-23.68	-138.74	74.13	21.40	14.80	-6.18	-6.24	-96.63	-42.24	6	4	41	48
1	7	27	[2.18]	-147.12	-147.12	152.13	-24.32	-142.81	76.06	21.21	14.74	-6.34	-6.38	-96.71	-44.40	6	4	41	48
1	7	28	[2.14]	-151.07	-151.07	156.07	-24.97	-146.84	78.04	21.91	15.39	-6.52	-6.53	-96.80	-46.16	6	5	42	49
1	8	29	[2.22]	-140.00	-140.00	145.00	-23.14	-135.33	72.50	18.43	11.41	-6.05	-6.11	-96.55	-41.16	6	4	40	48
1	8	30	[2.12]	-143.96	-143.96	148.96	-23.80	-139.88	74.48	21.53	14.10	-6.17	-6.26	-96.64	-46.48	6	5	44	47
1	8	31	[2.16]	-148.43	-148.43	153.43	-24.53	-144.19	76.71	21.02	14.69	-6.39	-6.43	-96.74	-45.87	6	4	42	48
1	8	32	[2.18]	-147.12	-147.12	152.13	-24.32	-142.81	76.06	21.21	14.74	-6.34	-6.38	-96.71	-44.40	6	4	41	48
1	9	33	[2.20]	-143.25	-143.25	148.25	-23.68	-138.74	74.13	21.40	14.80	-6.18	-6.24	-96.63	-42.24	6	4	41	48

Table 7.8: Optimisation process in financial trading strategy (continued)

TW	I	$Eval.$	X	$Fitness$	$Profit\ After\ C.$	MDD	ROI	$Profit$	$MDD\%$	MRU	$MRU\%$	$PROM$	RAR	RRR	SE	C_W	CL	W	L
1	9	34	[2.14]	-151.07	-151.07	156.07	-24.97	-146.84	78.04	21.91	15.39	-6.52	-6.53	-96.80	-46.16	6	5	42	49
1	9	35	[2.22]	-140.00	-140.00	145.00	-23.14	-135.33	72.50	18.43	11.41	-6.05	-6.11	-96.55	-41.16	6	4	40	48
1	9	36	[2.18]	-147.12	-147.12	152.13	-24.32	-142.81	76.06	21.21	14.74	-6.34	-6.38	-96.71	-44.40	6	4	41	48
1	10	37	[2.20]	-143.25	-143.25	148.25	-23.68	-138.74	74.13	21.40	14.80	-6.18	-6.24	-96.63	-42.24	6	4	41	48
1	10	38	[2.20]	-143.25	-143.25	148.25	-23.68	-138.74	74.13	21.40	14.80	-6.18	-6.24	-96.63	-42.24	6	4	41	48
1	10	39	[2.24]	-154.15	-154.15	159.15	-25.48	-150.11	79.58	18.81	11.64	-6.7	-6.65	-96.86	-45.25	6	5	36	49
1	10	40	[2.24]	-154.15	-154.15	159.15	-25.48	-150.11	79.58	18.81	11.64	-6.7	-6.65	-96.86	-45.25	6	5	36	49
1	11	41	[2.18]	-147.12	-147.12	152.13	-24.32	-142.81	76.06	21.21	14.74	-6.34	-6.38	-96.71	-44.40	6	4	41	48
1	11	42	[2.16]	-148.43	-148.43	153.43	-24.53	-144.19	76.71	21.02	14.69	-6.39	-6.43	-96.74	-45.87	6	4	42	48
1	11	43	[2.18]	-147.12	-147.12	152.13	-24.32	-142.81	76.06	21.21	14.74	-6.34	-6.38	-96.71	-44.40	6	4	41	48
1	11	44	[2.22]	-140.00	-140.00	145.00	-23.14	-135.33	72.50	18.43	11.41	-6.05	-6.11	-96.55	-41.16	6	4	40	48
1	12	45	[2.20]	-143.25	-143.25	148.25	-23.68	-138.74	74.13	21.40	14.80	-6.18	-6.24	-96.63	-42.24	6	4	41	48
1	12	46	[2.20]	-143.25	-143.25	148.25	-23.68	-138.74	74.13	21.40	14.80	-6.18	-6.24	-96.63	-42.24	6	4	41	48
1	12	47	[2.22]	-140.00	-140.00	145.00	-23.14	-135.33	72.50	18.43	11.41	-6.05	-6.11	-96.55	-41.16	6	4	40	48
1	12	48	[2.24]	-154.15	-154.15	159.15	-25.48	-150.11	79.58	18.81	11.64	-6.7	-6.65	-96.86	-45.25	6	5	36	49
1	13	49	[2.20]	-143.25	-143.25	148.25	-23.68	-138.74	74.13	21.40	14.80	-6.18	-6.24	-96.63	-42.24	6	4	41	48
1	13	50	[2.22]	-140.00	-140.00	145.00	-23.14	-135.33	72.50	18.43	11.41	-6.05	-6.11	-96.55	-41.16	6	4	40	48
1	13	51	[2.24]	-154.15	-154.15	159.15	-25.48	-150.11	79.58	18.81	11.64	-6.7	-6.65	-96.86	-45.25	6	5	36	49
1	13	52	[2.18]	-147.12	-147.12	152.13	-24.32	-142.81	76.06	21.21	14.74	-6.34	-6.38	-96.71	-44.40	6	4	41	48
1	14	53	[2.22]	-140.00	-140.00	145.00	-23.14	-135.33	72.50	18.43	11.41	-6.05	-6.11	-96.55	-41.16	6	4	40	48
1	14	54	[2.20]	-143.25	-143.25	148.25	-23.68	-138.74	74.13	21.40	14.80	-6.18	-6.24	-96.63	-42.24	6	4	41	48
1	14	55	[2.20]	-143.25	-143.25	148.25	-23.68	-138.74	74.13	21.40	14.80	-6.18	-6.24	-96.63	-42.24	6	4	41	48
1	14	56	[2.20]	-143.25	-143.25	148.25	-23.68	-138.74	74.13	21.40	14.80	-6.18	-6.24	-96.63	-42.24	6	4	41	48
1	15	57	[2.22]	-140.00	-140.00	145.00	-23.14	-135.33	72.50	18.43	11.41	-6.05	-6.11	-96.55	-41.16	6	4	40	48
1	15	58	[2.20]	-143.25	-143.25	148.25	-23.68	-138.74	74.13	21.40	14.80	-6.18	-6.24	-96.63	-42.24	6	4	41	48
1	15	59	[2.20]	-143.25	-143.25	148.25	-23.68	-138.74	74.13	21.40	14.80	-6.18	-6.24	-96.63	-42.24	6	4	41	48
1	15	60	[2.20]	-143.25	-143.25	148.25	-23.68	-138.74	74.13	21.40	14.80	-6.18	-6.24	-96.63	-42.24	6	4	41	48
1	16	61	[2.20]	-143.25	-143.25	148.25	-23.68	-138.74	74.13	21.40	14.80	-6.18	-6.24	-96.63	-42.24	6	4	41	48
1	16	62	[2.22]	-140.00	-140.00	145.00	-23.14	-135.33	72.50	18.43	11.41	-6.05	-6.11	-96.55	-41.16	6	4	40	48
1	16	63	[2.22]	-140.00	-140.00	145.00	-23.14	-135.33	72.50	18.43	11.41	-6.05	-6.11	-96.55	-41.16	6	4	40	48
1	16	64	[2.22]	-140.00	-140.00	145.00	-23.14	-135.33	72.50	18.43	11.41	-6.05	-6.11	-96.55	-41.16	6	4	40	48
1	17	65	[2.22]	-140.00	-140.00	145.00	-23.14	-135.33	72.50	18.43	11.41	-6.05	-6.11	-96.55	-41.16	6	4	40	48
1	17	66	[2.22]	-140.00	-140.00	145.00	-23.14	-135.33	72.50	18.43	11.41	-6.05	-6.11	-96.55	-41.16	6	4	40	48
1	17	67	[2.22]	-140.00	-140.00	145.00	-23.14	-135.33	72.50	18.43	11.41	-6.05	-6.11	-96.55	-41.16	6	4	40	48

Table 7.9: Optimisation process in financial trading strategy (continued)

TW	*I*	*Eval.*	*X*	*Fitness*	*Profit After C.*	*MDD*	*ROI*	*Profit*	*MDD%*	*MRU*	*MRU%*	*PROM*	*RAR*	*RRR*	*SE*	C_W	*CL*	*W*	*L*
1	17	68	[2.24]	-154.15	-154.15	159.15	-25.48	-150.11	79.58	18.81	11.64	-6.7	-6.65	-96.86	-45.25	6	5	36	49
1	18	69	[2.26]	-160.75	-160.75	165.75	-26.57	-156.79	82.87	19.19	11.87	-7.01	-6.89	-96.98	-46.35	6	5	34	50
1	18	70	[2.22]	-140.00	-140.00	145.00	-23.14	-135.33	72.50	18.43	11.41	-6.05	-6.11	-96.55	-41.16	6	4	40	48
1	18	71	[2.24]	-154.15	-154.15	159.15	-25.48	-150.11	79.58	18.81	11.64	-6.7	-6.65	-96.86	-45.25	6	5	36	49
1	18	72	[2.24]	-154.15	-154.15	159.15	-25.48	-150.11	79.58	18.81	11.64	-6.7	-6.65	-96.86	-45.25	6	5	36	49
1	19	73	[2.22]	-140.00	-140.00	145.00	-23.14	-135.33	72.50	18.43	11.41	-6.05	-6.11	-96.55	-41.16	6	4	40	48
1	19	74	[2.26]	-160.75	-160.75	165.75	-26.57	-156.79	82.87	19.19	11.87	-7.01	-6.89	-96.98	-46.35	6	5	34	50
1	19	75	[2.22]	-140.00	-140.00	145.00	-23.14	-135.33	72.50	18.43	11.41	-6.05	-6.11	-96.55	-41.16	6	4	40	48
1	19	76	[2.22]	-140.00	-140.00	145.00	-23.14	-135.33	72.50	18.43	11.41	-6.05	-6.11	-96.55	-41.16	6	4	40	48
1	20	77	[2.22]	-140.00	-140.00	145.00	-23.14	-135.33	72.50	18.43	11.41	-6.05	-6.11	-96.55	-41.16	6	4	40	48
1	20	78	[2.20]	-143.25	-143.25	148.25	-23.68	-138.74	74.13	21.40	14.80	-6.18	-6.24	-96.63	-42.24	6	4	41	48
1	20	79	[2.24]	-154.15	-154.15	159.15	-25.48	-150.11	79.58	18.81	11.64	-6.7	-6.65	-96.86	-45.25	6	5	36	49
1	20	80	[2.20]	-143.25	-143.25	148.25	-23.68	-138.74	74.13	21.40	14.80	-6.18	-6.24	-96.63	-42.24	6	4	41	48
1	21	81	[2.22]	-140.00	-140.00	145.00	-23.14	-135.33	72.50	18.43	11.41	-6.05	-6.11	-96.55	-41.16	6	4	40	48
1	21	82	[2.22]	-140.00	-140.00	145.00	-23.14	-135.33	72.50	18.43	11.41	-6.05	-6.11	-96.55	-41.16	6	4	40	48
1	21	83	[2.24]	-154.15	-154.15	159.15	-25.48	-150.11	79.58	18.81	11.64	-6.7	-6.65	-96.86	-45.25	6	5	36	49
1	21	84	[2.24]	-154.15	-154.15	159.15	-25.48	-150.11	79.58	18.81	11.64	-6.7	-6.65	-96.86	-45.25	6	5	36	49
1	22	85	[2.22]	-140.00	-140.00	145.00	-23.14	-135.33	72.50	18.43	11.41	-6.05	-6.11	-96.55	-41.16	6	4	40	48
1	22	86	[2.22]	-140.00	-140.00	145.00	-23.14	-135.33	72.50	18.43	11.41	-6.05	-6.11	-96.55	-41.16	6	4	40	48
1	22	87	[2.22]	-140.00	-140.00	145.00	-23.14	-135.33	72.50	18.43	11.41	-6.05	-6.11	-96.55	-41.16	6	4	40	48
1	22	88	[2.22]	-140.00	-140.00	145.00	-23.14	-135.33	72.50	18.43	11.41	-6.05	-6.11	-96.55	-41.16	6	4	40	48
1	23	89	[2.22]	-140.00	-140.00	145.00	-23.14	-135.33	72.50	18.43	11.41	-6.05	-6.11	-96.55	-41.16	6	4	40	48
1	23	90	[2.20]	-143.25	-143.25	148.25	-23.68	-138.74	74.13	21.40	14.80	-6.18	-6.24	-96.63	-42.24	6	4	41	48
1	23	91	[2.22]	-140.00	-140.00	145.00	-23.14	-135.33	72.50	18.43	11.41	-6.05	-6.11	-96.55	-41.16	6	4	40	48
1	23	92	[2.24]	-154.15	-154.15	159.15	-25.48	-150.11	79.58	18.81	11.64	-6.7	-6.65	-96.86	-45.25	6	5	36	49
1	24	93	[2.22]	-140.00	-140.00	145.00	-23.14	-135.33	72.50	18.43	11.41	-6.05	-6.11	-96.55	-41.16	6	4	40	48
1	24	94	[2.22]	-140.00	-140.00	145.00	-23.14	-135.33	72.50	18.43	11.41	-6.05	-6.11	-96.55	-41.16	6	4	40	48
1	24	95	[2.24]	-154.15	-154.15	159.15	-25.48	-150.11	79.58	18.81	11.64	-6.7	-6.65	-96.86	-45.25	6	5	36	49
1	24	96	[2.22]	-140.00	-140.00	145.00	-23.14	-135.33	72.50	18.43	11.41	-6.05	-6.11	-96.55	-41.16	6	4	40	48
1	25	97	[2.24]	-154.15	-154.15	159.15	-25.48	-150.11	79.58	18.81	11.64	-6.7	-6.65	-96.86	-45.25	6	5	36	49
1	25	98	[2.22]	-140.00	-140.00	145.00	-23.14	-135.33	72.50	18.43	11.41	-6.05	-6.11	-96.55	-41.16	6	4	40	48
1	25	99	[2.20]	-143.25	-143.25	148.25	-23.68	-138.74	74.13	21.40	14.80	-6.18	-6.24	-96.63	-42.24	6	4	41	48
1	25	100	[2.24]	-154.15	-154.15	159.15	-25.48	-150.11	79.58	18.81	11.64	-6.7	-6.65	-96.86	-45.25	6	5	36	49
1	26	101	[2.22]	-140.00	-140.00	145.00	-23.14	-135.33	72.50	18.43	11.41	-6.05	-6.11	-96.55	-41.16	6	4	40	48

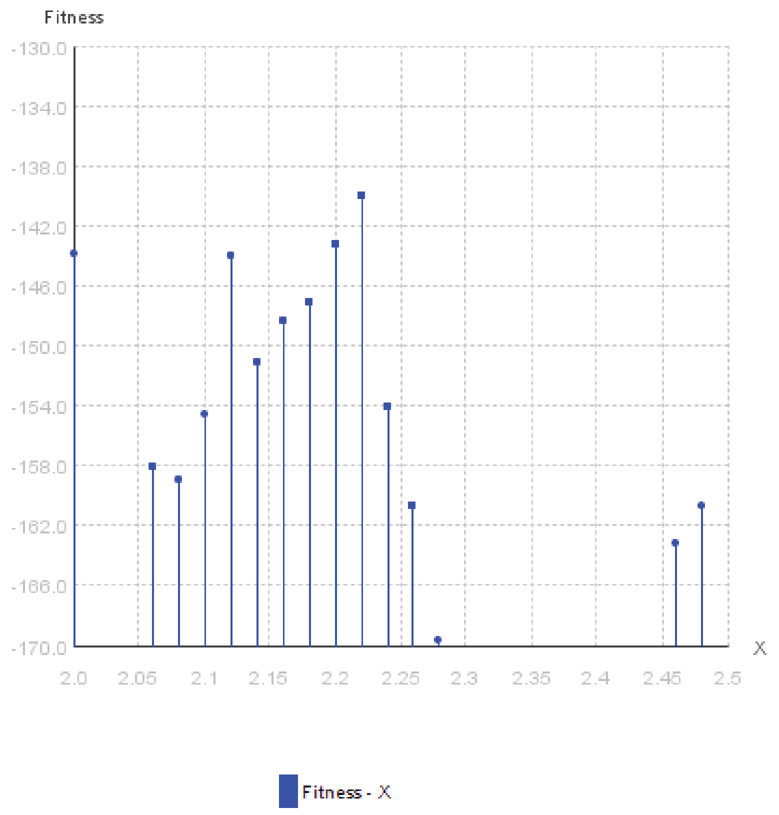


Figure 7.10: Optimisation process in financial trading strategy

7.4 Football Strategies

This section will explain and evaluate different football strategies at increasing levels of complexity. Each strategy will attempt to build on from the previous to demonstrate an additional feature of UBEL. All the strategies presented in this section were tested on 51 Football matches.

7.4.1 Back the Home Team

Strategy: *"As soon as the football match goes in-play, back the home team to win for £10 and then exit"*. Table 7.10 summarises the results.

```
function backTheHomeTeam(var sportingEvent,var stake){
  var strategyTag = Tag.new;
  var bet=[];
  var EXCHANGE = "Betfair";
  var matchOdds = sportingEvent.market("Match Odds");
  strategyTag:{
       at(matchOdds.inplay){
                 var backBet = Back.new("Match Odds",sportingEvent.homeTeam.name,stake,EXCHANGE);
                 bet = sportingEvent.placeBet([backBet]);
                 strategyTag.stop;
              };
        };
  };
};
```

Table 7.10: Results from the “Back the Home Team” strategy in Football

Profit (£)	-26.53
Profit after commission (£)	-37.32
MDD (£)	155.80
MDD (%)	55.07%
MRU (£)	92.93
MRU (%)	48.91%
ROI (%)	-7.43%
PROM (%)	-1.45%
RRR (%)	-23.95%
RAR (%)	-1.61%
Perfect Profit (£)	627.88
Total Winnings (£)	205.01
Total Loses (£)	-242.33
#Number of wins	26
#Number of loses	25
#Maximum number of consecutive wins	5
#Maximum number of consecutive loses	2
Strategy Efficiency	-5.94%

This strategy is not profitable. It is also interesting that we won 50.98% of our bets and still losing money. Let us refine our strategy by instead of placing a bet on the market favourite.

7.4.2 Back the Favourite

```
function backTheFavourite(var sportingEvent,var stake){
  var strategyTag = Tag.new;
  var bet=[];
  var EXCHANGE = "Betfair";
  var matchOdds = sportingEvent.market("Match Odds");
  strategyTag:{
       at(matchOdds.inplay){
                 var backBet = Back.new("Match Odds",matchOdds.favourite,stake,EXCHANGE);
                 bet = sportingEvent.placeBet([backBet]);
                 strategyTag.stop;
              };
        };
  };
};
```

Table 7.11 summarises the results.

Table 7.11: Results from the "Back the Favourite" strategy in Football

Profit (£)	3.9
Profit after commission (£)	-6.79
MDD (£)	170.65
MDD (%)	65.58%
MRU (£)	108.45
MRU (%)	121.06%
ROI (%)	-1.33%
PROM (%)	0.10%
RRR (%)	-3.98%
RAR (%)	-0.29%
Perfect Profit (£)	369.17
Total Winnings (£)	203.21
Total Loses (£)	-210
#Number of wins	30
#Number of loses	21
#Maximum number of consecutive wins	8
#Maximum number of consecutive loses	2
Strategy Efficiency	-1.84%

This strategy is also not profitable, but this is because of the commission. However, in comparison to the previous strategy, this one performs better.

7.4.3 Lay The Draw

The following strategy will only place a lay bet on the draw. So the strategy will be: *"1 minute before the match start, if in the market 'Match Odds' the lay odds of the selection 'The Draw' are less or equal than 4 and more or equal than 2, then lay the draw in the market 'Match Odds' at the best available odds with the given stake"*. The input of the strategy is the sporting event and the stake. The initial stake of the lay bet was £20. Table 7.12 summarises the results.

```
 function layTheDraw(var sportingEvent,var stake){
   var strategyTag = Tag.new;
   var bet=[];
   var EXCHANGE = "Betfair";
   var matchOdds = sportingEvent.market("Match Odds");
   var theDraw = sportingEvent.market("Match Odds").selection("The Draw");
   strategyTag:{
        at(sportingEvent.clock >= sportingEvent.startTime - 1min){
               if(theDraw.lp[0]<=4 && theDraw.lp[0]>=2){
                  var layBet = Lay.new("Match Odds","The Draw",stake,EXCHANGE);
                  bet = sportingEvent.placeBet([layBet]);
                  strategyTag.stop;
               };
         };
   };
};
```

Table 7.12: Results from the "Lay the Draw" strategy in Football

Profit (£)	131.37
Profit after commission (£)	117.05
MDD (£)	232
MDD (%)	78.64%
MRU (£)	290
MRU (%)	460.32%
ROI (%)	12.39%
PROM (%)	6.81%
RRR (%)	50.45%
RAR (%)	4.75%
Perfect Profit (£)	329.05
Total Winnings (£)	272.05
Total Loses (£)	-155
#Number of wins	15
#Number of loses	3
#Maximum number of consecutive wins	5
#Maximum number of consecutive loses	3
Strategy Efficiency	35.57%

This strategy gives us a small profit. Consider a slight modification to the strategy.

7.4.4 Advanced Lay The Draw

"1 minute before the match start, if in the market 'Match Odds' the lay odds of the selection 'The Draw' are less or equal than 4 and more or equal than 2 then:

1. *Lay the draw in the market 'Match Odds' at the best available odds with the given stake*
2. *If the lay bet is matched and a goal is being detected :*
   - *If is the first goal in the match and the match is less than 100 minutes in-play then calculate the minimum odds to back the selection 'The Draw' in order to have a profit no matter what the result is.*

- *If is not the first goal then : If the difference of home goals with away goals is more than 1 then if the back bet from step 4 is placed and is not partial matched, cancel it*

3. *When the odds of 'The Draw' reach the price from step 2, calculate the back stake to spread the profit across all the selections in the market 'Match Odds'.*

4. *Place the back bet at the best available odds with the stake calculated from step 3.*

5. *When the bet from step 4 is matched exit the strategy".*

The inputs of the strategy were the football match, the initial lay stake (£20) and the minimum percent we wanted to win from the initial lay stake (60%). Table 7.13 summarises the results.

Table 7.13: Results from the advanced "Lay the Draw" strategy in Football

Profit (£)	371.89
Profit after commission (£)	350.6
MDD (£)	475.09
MDD (%)	149.67%
MRU (£)	708.27
MRU (%)	549.21%
ROI (%)	13.51%
PROM (%)	15.97%
RRR (%)	73.80%
RAR (%)	11.88%
Perfect Profit (£)	437.65
Total Winnings (£)	404.46
Total Loses (£)	-53.87
#Number of wins	34
#Number of loses	2
#Maximum number of consecutive wins	12
#Maximum number of consecutive loses	2
Strategy Efficiency	80.11%

The results show that the modified "Lay The Draw" strategy performs much better than the original one.

```
function layTheDraw(var sportingEvent,var stake,var percent){
 var strategyTag = Tag.new;
 var bet=[];
 var bet2=[];
 var bet3=[];
 var EXCHANGE = "Betfair";
 var matchOdds = sportingEvent.market("Match Odds");
 var theDraw = sportingEvent.market("Match Odds").selection("The Draw");
 strategyTag:{
   at(sportingEvent.clock >= sportingEvent.startTime - 1min){
     if(theDraw.lp[0]<=4 && theDraw.lp[0]>=2){
       var layBet = Lay.new("Match Odds","The Draw",stake,EXCHANGE);
       bet = sportingEvent.placeBet([layBet]);
       at(bet[0].matched){
         var firstGoal = false;
         var firstTag = Tag.new;
         at(sportingEvent.goal?(var team)){
           if(sportingEvent.getInplayTime()<=100min){
             if(!firstGoal){
               firstGoal = true;
               firstTag :{
                 var betOdds = greenOdds("back",percent,bet[0].odds,stake,EXCHANGE);
                 var oddsAccepted = false;
                 at(matchOdds.marketUpdated?){
                   if(!oddsAccepted){
                     if(theDraw.bp[0]>=betOdds){
                       oddsAccepted = true;
                       var bp = theDraw.bp[0];
                       var backStake = greenStake(bet[0].odds,stake,bp,EXCHANGE);
                       var backBet = Back.new("Match Odds","The Draw",bp,backStake,EXCHANGE);
                       bet3 = sportingEvent.placeBet([backBet]);
                       at(bet3[0].matched){
                         firstTag.stop;
                         strategyTag.stop;
             };};};};};},}
             else{
               firstTag.freeze;
               if(abs(sportingEvent.homeGoals - sportingEvent.awayGoals)>=2){
                 if(!bet3.empty){
                   if(!bet3[0].partialMatch){
                     bet3[0].cancelBet();
                   };};}
               else{
                 firstTag.unfreeze;
               };};};};};};}
       else{
         strategyTag.stop;
 };};};};
```

Figure 7.11 shows the strategy and market visualisation for the match Paris St-G and Barcelona 2013 for the UEFA Champions League.

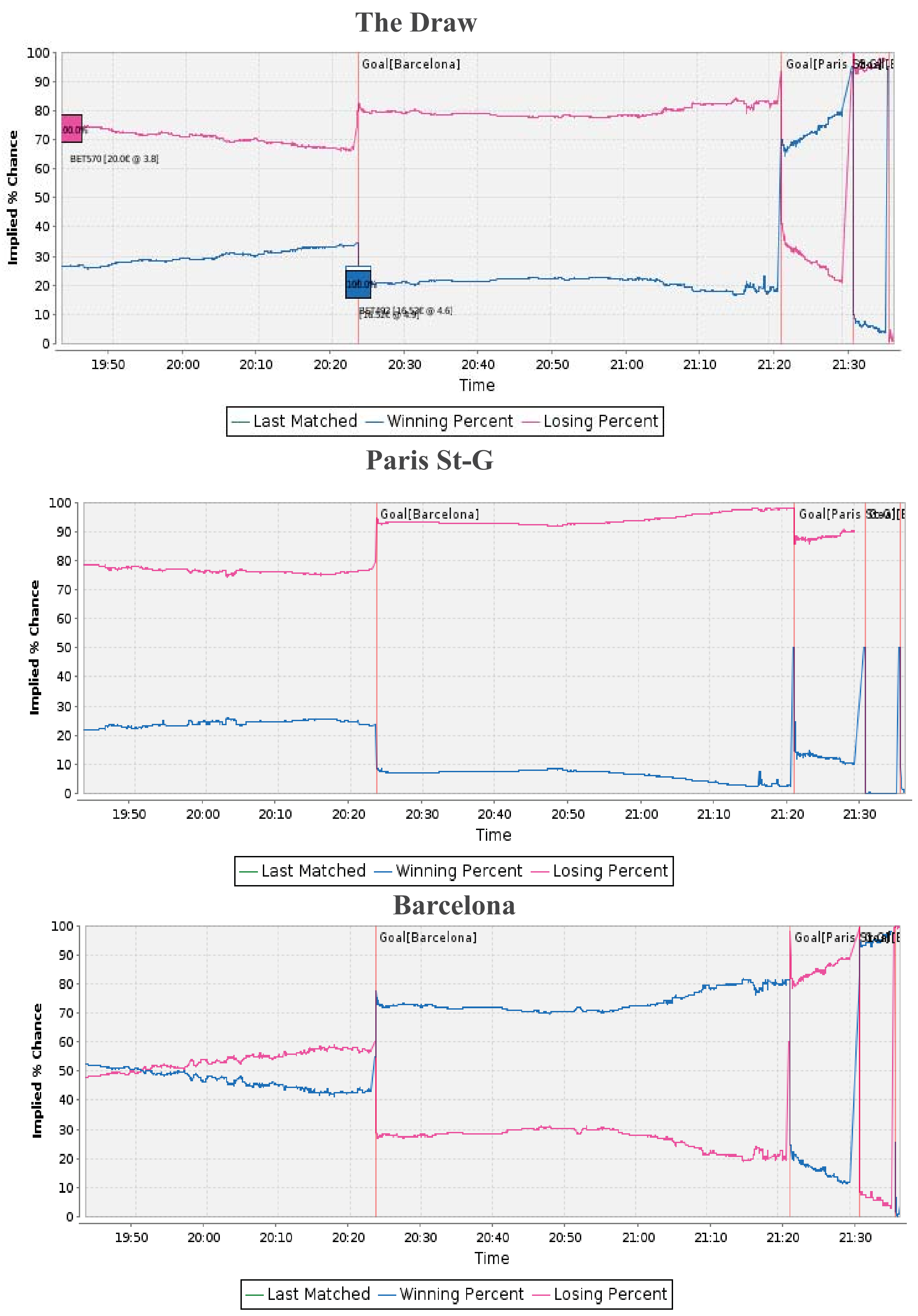


Figure 7.11: Visualisation of the "Lay the Draw" strategy and odds evolution in the Match Odds market over the course of a single match (Paris St-G v Barcelona).

Chapter 8

Conclusion

8.1 Summary of Achievements

This thesis has addressed the challenges related to the generic specification, back- testing, optimisation and execution of parameterised automated trading strategies for betting exchange markets.

Over the past few years, betting exchanges have been attracting the interest of researchers at a growing pace, not only due to their economic importance but also to the similarities with traditional financial markets.

There are many sophisticated automated trading software tools in finance, which traders can use to test and execute their strategies, or even test their quantitative models. Yet to our knowledge there is no publicly available quantitative trading research platform for sports betting exchange trading strategies that alleviates traders from coding up those "infrastructural" components. Any prior work in this area has mainly been focused on predicting the outcome in a market. This thesis rectifies this by presenting a related framework called SPORTSBET. The work presented is innovative and expands the applicability, capacity and specification power of prior work in the research areas of finance, betting and artificial intelligence.

One of the main key contributions of this thesis is the betting strategy programming language

(UBEL) described in Chapter 3. UBEL is built on top of UrbiScript, which is a dynamic script language used in robotics and artificial intelligence. It supports and emphasizes events programming and parallelism, two major paradigms in complex systems programming, by offering language constructs and primitives. UBEL allows users to specify betting exchange trading strategies in a rigorous manner and is powerful and generic enough to support a wide range of sports related to betting exchange markets. Moreover, UBEL as shown in Chapter 7, simplifies the process of writing both simple and complex strategies for betting exchange markets.

The complex event processor presented in Chapter 3 is the second major contribution of this work. SPORTSBET engine is capable of synchronising multiple real time data streams and replaying them on an historical basis with dynamic market re-construction. This engine can process large volumes of incoming streams or events, regardless of whether incoming streams are historical or real-time in nature. The engine filters and analyses events in various ways, and responds to conditions of interest with minimal latency. This complex event processor supports the parallel simulation and/or live monitoring of many different sporting events and their associated markets and can place or cancel many bets in parallel across different markets.

The third major contribution is the optimisation platform presented in Chapter 5 whereby strategy parameters are automatically refined using a stochastic search heuristic in order to improve strategy performance. The method and the metrics adopted were inspired from the financial sector.

Chapter 6 describes the final toolset which has: a simple, intuitive and responsive user interface, ultimately minimising the effort and time consumed to perform a desired task, an editor designed specific for UBEL which allows users writing their UBEL scripts efficiently and finally a database management system which can be used to perform research and analysis, to test out potential betting systems and strategies, and to build predictive models or ratings to assess comparative chances within a sporting event.

Finally, in Chapter 7 we have presented numerical results produced using SPORTSBET framework.

8.2 Future Work

There are a number of possible extensions to the work presented in this thesis. By extending the UBEL language one can cover a wider range of sports or even different areas such as the financial sector. Another area of future research could be the implementation of different search algorithms used during the optimisation process, and the development of a benchmarking tool where the user can find individual performance data on the behaviour of each algorithm in order to evaluate the algorithm performance. Similarly, the implementation of different strategy evaluation techniques and their benchmarking. SPORTSBET at the moment can detect in-play events such as goal being scored, by analysing the price movements across different markets related to a specific sporting event. It would be very interesting to detect those events by analysing the price movements from a single market in addition to detect events not supported at the moment such as a red card given. SPORTSBET during the back-testing process assumes the trader will place small bets that can't affect the market, so an extension could be the development of an impact model which can be used to examine the effect of a strategy on the market.

Currently, each sporting event's market is associated with only one betting exchange provider. So for each selection in a market there is a matrix with the back prices, a matrix with the back volumes, a matrix with the lay prices and a matrix with the lay volumes. It would be more advanced and beneficial if for each market we could have the prices from different betting providers under the same matrix (see Figure 8.1).

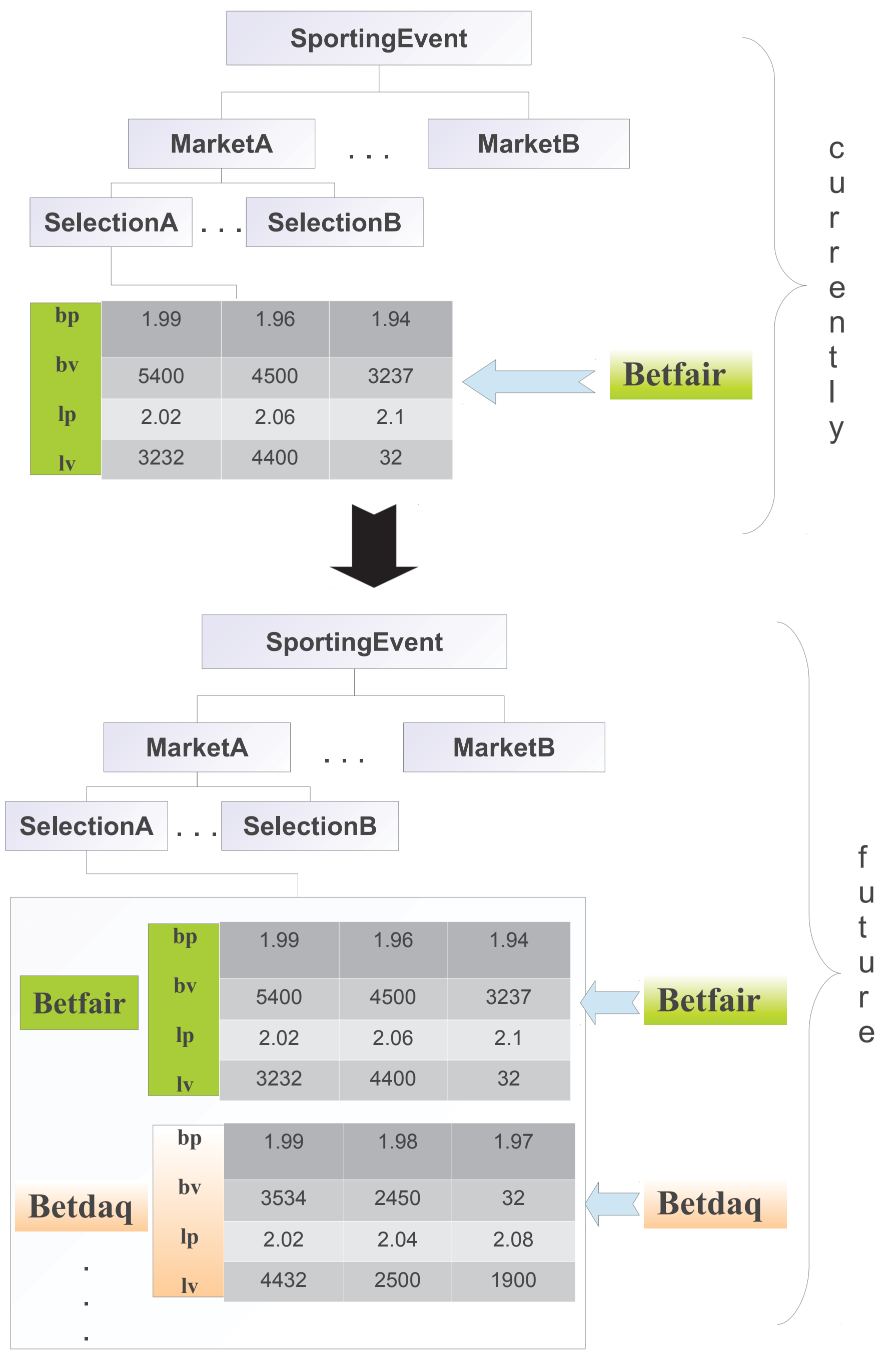


Figure 8.1: Future selection's matrix prices of UBEL

Finally, it would be interesting to investigate the idea of the automatic evolution of entirely new betting strategies using genetic algorithms.

Appendix A

UBEL functions

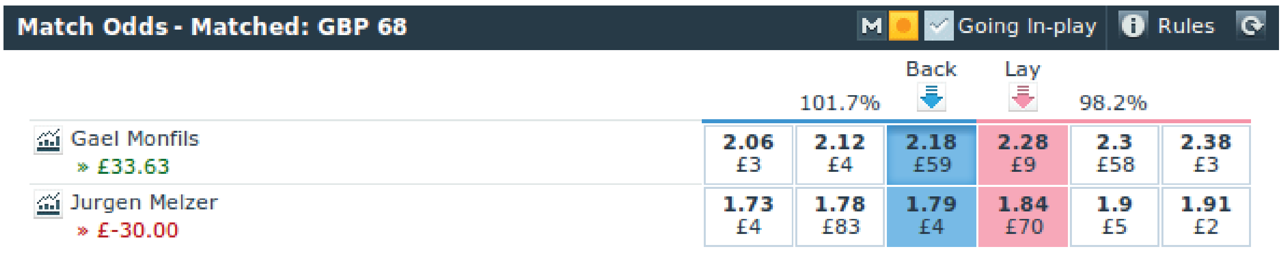
A.1 Hedging Functions

spreadProfit : Similar with the function `greenStake` with the difference that is working even if you have many bets matched in the market.

Input Parameters :

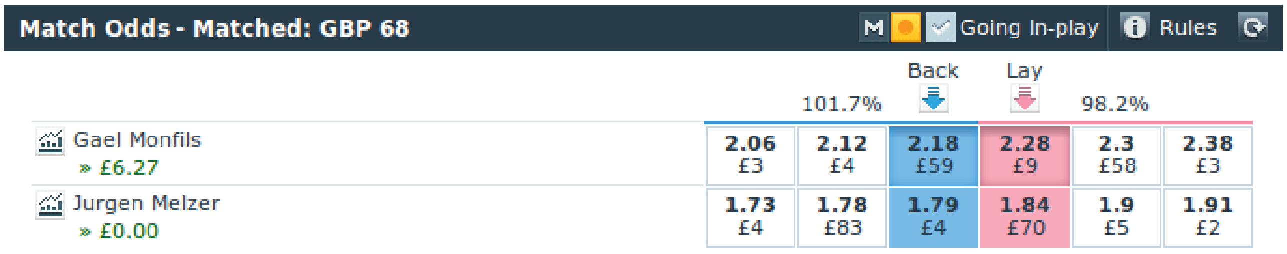
- The selection object.
- The bet type (Back or Lay).
- The name of the exchange.

greenOdds : Calculate the minimum odds to back or the maximum odds to lay a selection, in order to have a profit equal to or more than the input percentage of the initial stake, no matter what the result is. Suppose we have a tennis match between Gael Monfils and Jurgen Melzer. Suppose we back Monfils @ 2.18 (the best backing odds available now) with £30.


Match Odds - Matched: GBP 68

			Back	Lay		
	101.7%				98.2%	
Gael Monfils » £33.63	2.06 £3	2.12 £4	2.18 £59	2.28 £9	2.3 £58	2.38 £3
Jurgen Melzer » £-30.00	1.73 £4	1.78 £83	1.79 £4	1.84 £70	1.9 £5	1.91 £2

So under Monfils we have the sum we would win if we left the bet like that until the end and Monfils wins the match. On Melzer is the sum we will lose if Melzer wins the match. So if we want to know which is the maximum lay price on Monfils in order to win 20% from our initial stake. We can use the function `greenOdds`. And the output will be 1.96. So if we lay Monfils @ 1.96 for £30 we will have:

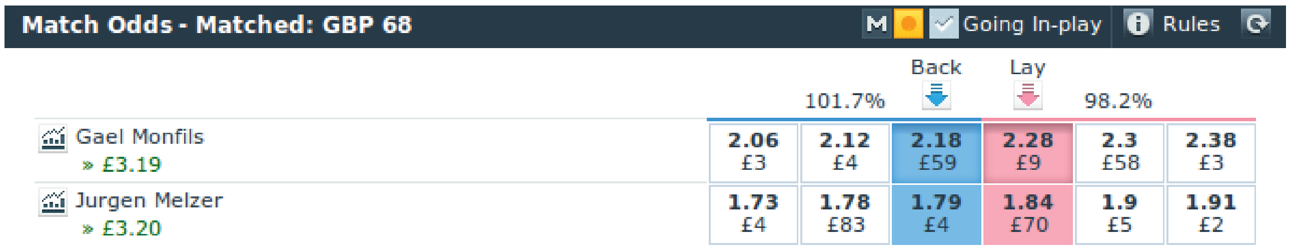

Match Odds - Matched: GBP 68

			Back	Lay		
	101.7%				98.2%	
Gael Monfils » £6.27	2.06 £3	2.12 £4	2.18 £59	2.28 £9	2.3 £58	2.38 £3
Jurgen Melzer » £0.00	1.73 £4	1.78 £83	1.79 £4	1.84 £70	1.9 £5	1.91 £2

<u>Input Parameters</u> :

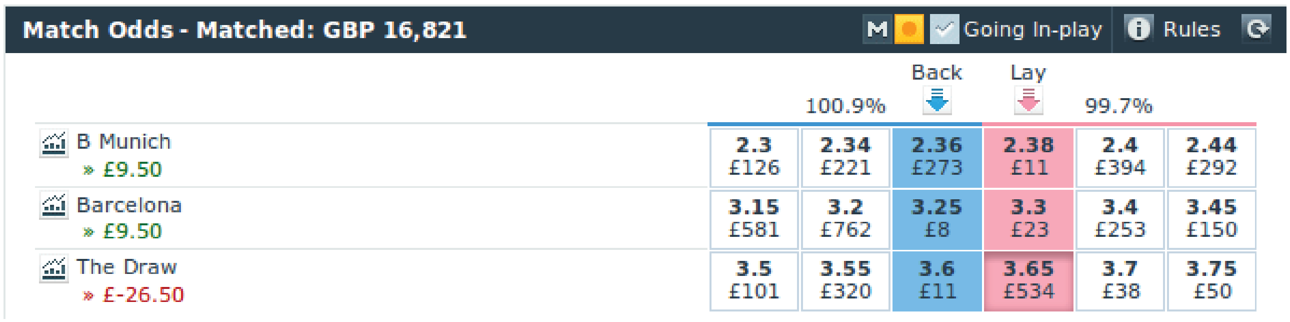
- The bet type (Back or Lay).
- The percentage of the initial stake we want to win.
- The odds of the initial bet.
- The stake of the initial bet.
- The name of the exchange.

Moreover, if you want to spread the profit across Monfils and Melzer you can use `greenStake` or `spreadProfit` and the result will be to lay Monfils @ 1.96 for £33.37.


Match Odds - Matched: GBP 68

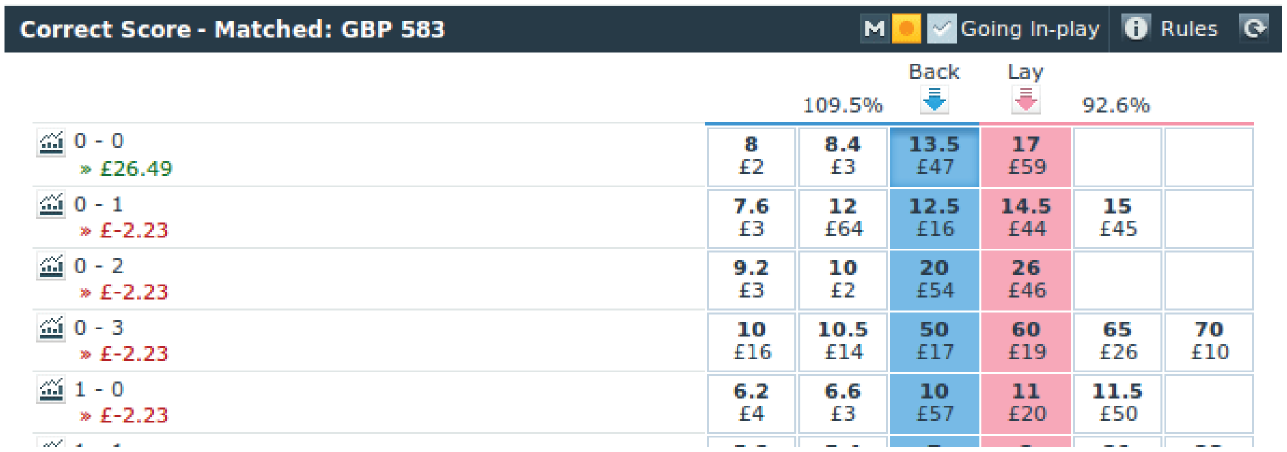
	101.7%		Back	Lay	98.2%	
Gael Monfils » £3.19	2.06 £3	2.12 £4	2.18 £59	2.28 £9	2.3 £58	2.38 £3
Jurgen Melzer » £3.20	1.73 £4	1.78 £83	1.79 £4	1.84 £70	1.9 £5	1.91 £2

balanceStake : Calculates the stake to place a back or lay bet, at the given odds, in order to balance the losses from other matched bets. Suppose we have a football match between Barcelona and B. Munich. Suppose we lay The Draw @ 3.65 with £10. So under B. Munich and Barcelona we have the sum we would win if we left the bet like that until the end and one of these selections is the winner. On The Draw is the sum we will lose if The Draw is the winner.


Match Odds - Matched: GBP 16,821

	100.9%		Back	Lay	99.7%	
B Munich » £9.50	2.3 £126	2.34 £221	2.36 £273	2.38 £11	2.4 £394	2.44 £292
Barcelona » £9.50	3.15 £581	3.2 £762	3.25 £8	3.3 £23	3.4 £253	3.45 £150
The Draw » £-26.50	3.5 £101	3.55 £320	3.6 £11	3.65 £534	3.7 £38	3.75 £50

Suppose we want to cover the risk of the football match ends 0 - 0. So what we want to do is back the selection 0 - 0 at the best available odds with a stake balancing the liability in the market Match Odds. So using the function `balanceStake` we back in the market Correct Score the selection 0 - 0 @ 13.5 for £2.23.



Correct Score - Matched: GBP 583 — Going In-play — Rules

		109.5%	Back	Lay	92.6%	
0 - 0 » £26.49	8 £2	8.4 £3	13.5 £47	17 £59		
0 - 1 » £-2.23	7.6 £3	12 £64	12.5 £16	14.5 £44	15 £45	
0 - 2 » £-2.23	9.2 £3	10 £2	20 £54	26 £46		
0 - 3 » £-2.23	10 £16	10.5 £14	50 £17	60 £19	65 £26	70 £10
1 - 0 » £-2.23	6.2 £4	6.6 £3	10 £57	11 £20	11.5 £50	

Input Parameters :

- The bet type (Back or Lay).
- The result in case of winning and losing of a market
- The odds of the bet you want to place.
- The name of the exchange.

A.2 Odds Control functions

getNextLayOdds : Returns the next acceptable lay price which is after the given one.

Input Parameters :

- The input price.
- The name of the exchange.

ticksDiff : Returns difference in ticks between two given prices. The result is positive if the first input price is lower from the second one and negative if it is higher.

Input Parameters :

- The first input price.

- The second input price.

- The name of the exchange.

getNearestOdds : Returns the closest acceptable price from the given one.

Input Parameters :

- The input price.

- The name of the exchange.

A.3 Dutching functions

rangeDutch : Same as dutch, with the difference that you back a range of selections. Ex: [1,4] will back the selections 1, 2 and 3. The selections are sorted with 0 being the favourite.

Input Parameters :

- The sporting event.

- The market name.

- The range of selections.

- The stake.

- The name of the exchange.

Reverse dutch betting is the same as dutching with the difference that is laying on more than one outcome.

reverseDutch : Laying the given selections of the given market and sporting event for the given stake. If the stake needed to lay all the given selections is more than the input stake, no bet is placed.

Input Parameters :

- The sporting event.
- The market name.
- The selections names.
- The stake.
- Boolean value. If false the stake will be considered as the maximum liability, else as the maximum payout.
- The name of the exchange.

reverseRangeDutch : Same as reverse dutch, with the difference that you lay a range of selections. Ex: [1,4] will lay the selections 1, 2 and 3. The selections are sorted with 0 being the favourite.

Input Parameters :

- The sporting event.
- The market name.
- The range of selections.
- The stake.
- Boolean value. If false the stake will be considered as the maximum liability, or else as the maximum payout.
- The name of the exchange.

A.4 Execution functions

simulate : Start simulating all the available markets of all the sporting events in a SportingEvent-Collector.

Input Parameters :

- **when** : At what historical time the simulation of the sporting events will start, using as base time the historical start time of the event.
- **collector** : The `SportingEventCollector` object.

waitResult : Stop monitoring the market and wait for the result. Input Parameters :

- **market** : The id of the market.
- **exchange** : The name of the exchange.

startOptimisation : Start the optimisation of a strategy. Input Parameters :

- **collector** : The sportingEventCollector object.
- **when** : At what historical time the simulation of the sporting events will start, using as base time the historical start time of the event.
- **strategy** : The strategy name.
- **fitness** : The fitness function to be used.
- **values** : An array which contains values for the optimisation algorithm. Such as constants, standard deviation, the dimension of the problem etc.
- **constrain** : The definition of the constrain function.

Appendix B

Betfair API services and Price Increments

Since the decimal odds requested for any bet, back or lay, require the exact price format and range used by Betfair, automated programs must be capable of specifying the values in this range. The below table show the price increments for Betfair odds, as discussed in Chapter 3.

Table B.1: Price Increments for Betfair Odds Markets

Decimal Odds Range	**Increment**
$1.01 \rightarrow 2$	0.01
$2 \rightarrow 3$	0.02
$3 \rightarrow 4$	0.05
$4 \rightarrow 6$	0.1
$6 \rightarrow 10$	0.2
$10 \rightarrow 20$	0.5
$20 \rightarrow 30$	1
$30 \rightarrow 50$	2
$50 \rightarrow 100$	5
$100 \rightarrow 1000$	10

In a UBEL data structure, this can be represented as a list comprising of the following values (only the first few values are shown below). Such an array can be used to automatically increment or decrement a price, as shown in Chapter 3, by using the appropriate functions.

```
Betfair acceptable odds in UBEL
var Betfair.acceptableOdds =  [1.01, 1.02, 1.03, 1.04, 1.05, 1.06, 1.07,

                              1.08, 1.09, 1, 1.1, 1.11, 1.12, 1.13, ... 1000];


getNextBackOdds(1.023, "Betfair");

*** 1.03


getNextLayOdds(1.01, "Betfair");

*** 1.01


getCloserOdds(1.041, "Betfair");

*** 1.04


ticks(1.02, 5, "Betfair");

*** 1.07
```

Tables B.2, B.3 and B.4 below lists all Betfair API Services, or calls which can be used for automating betting strategies, with corresponding access levels for each depending upon the user's subscription level, within SPORTSBET. The most commonly used services are described in Chapter 6.

Table B.2: Listing of Betfair API Services

Betfair API Service Name	Description	Full Access	Free
Login	Log in to the API service and initiate a secure session	✓	24 p/m
Logout	Explicitly end your session	✓	✓
KeepAlive	Stop a session timing out	✓	✓
ConvertCurrency	Convert a currency, based on the Betfair currency exchange rate	✓	✗
GetActiveEventTypes	Retrieve lists of all categories of sporting events	✓	✓
GetAllCurrencies	Retrieve all the currencies	✓	✗
GetAllEventTypes	Retrieve lists of all categories of sports that have at least one market associated with them	✓	✓
GetAllMarkets	Retrieve information about all of the markets that are currently active or suspended on the given exchange	✓	✓
GetBet	Retrieve information about a particular bet	✓	60 p/m
GetBetHistory	Retrieve information about the bets you have placed on a particular exchange	✓	1 p/m
GetBetLite	Retrieve information about a bet. This is the lite version of the GetBet service which returns less information for the bet	✓	60 p/m
GetBetMatchesLite	Retrieve information about a the matched portion of a bet	✓	60 p/m
GetCompleteMarketPricesCompressed	Retrieve all back and lay stakes for each price on the exchange for a given Market ID in a compressed format	✓	60 p/m
GetCurrentBets	Retrieve information about your current bets on a particular exchange server	✓	60 p/m
GetSilks and GetSilks2	Retrieve a relative URL to the jockey silk image and data about each selection	✓	✗

Table B.3: Listing of Betfair API Services (continued)

Betfair API Service Name	Description	Full Access	Free
GetCurrentBetsLite	Retrieve information about your current bets on a particular exchange	✓	60 p/m
GetDetailAvailableMarketDepth	Returns the current odds and available back/lay amounts on a runner in an event	✓	60 p/m
GetEvents	Navigate through the events hierarchy until you reach details of the betting market for an event that you are interested in	✓	✓
GetInPlayMarkets	Retrieve the markets that will be turned in-play in the next 24 hours	✓	✗
GetMarket	Retrieve all static market data for a given Market ID	✓	5 p/m
GetMarketPrices	Retrieve dynamic market data for a given Market ID	✓	10 p/m
GetMarketPricesCompressed	Retrieve dynamic market data for a given Market ID in a compressed format	✓	60 p/m
GetMUBets	Retrieve information about all your matched and unmatched bets on a particular exchange server	✓	60 p/m
GetMUBetsLite	Retrieve information about all your matched and unmatched bets on a particular exchange server	✓	60 p/m
GetMarketProfitAndLoss	Retrieve Profit and Loss information for the user account in a given market	✓	60 p/m
GetMarketTradedVolume	Obtain all the current odds and matched amounts on a single runner in a particular event	✓	60 p/m
GetMarketTradedVolumeCompressed	Obtain the current price and matched amounts at each price on all of the runners in a particular market	✓	60 p/m
GetPrivateMarkets	Retrieve active and suspended private markets that are within an EventType that is not visible on Betfair.com or with the GetEvents or GetActiveEvents services	✓	✓

Table B.4: Listing of Betfair API Services (continued)

Betfair API Service Name	**Description**	**Full Access**	**Free**
CancelBets	Cancel multiple unmatched (1 to 40) bets placed on a single Market	✓	✓
CancelBetsByMarket	Cancel all unmatched bets (or unmatched portions of bets) placed on one or more Markets	✓	✗
PlaceBets	Place multiple (1 to 60) bets on a single Market	✓	100 p/m
UpdateBets	Edit multiple (1 to 15) bets on a single Market	✓	✓
GetAccountFunds	Retrieve information about your local wallet on a particular exchange server	✓	12 p/m
DepositFromPaymentCard	Deposit funds into your UK wallet from a previously registered payment card	✓	✗
GetAccountStatement	Obtain information about transactions involving your local wallet on an exchange server	✓	1 p/m
TransferFunds	Transfer funds between your UK and Australian account wallets	✓	✓

p/m = per minute

Bibliography

[1] R. Adaikkalavan and S. Chakravarthy. SnoopIB: Interval-based Event Specification and Detection for Active Databases. *Data and Knowledge Engineering*, 59(1):139–165, 2006.

[2] J. Agrawal, Y. Diao, D. Gyllstrom, and N. Immerman. Efficient Pattern Matching Over Event Streams. In *Proc. of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 147–160. ACM, 2008.

[3] E. Alpaydin. *Introduction to Machine Learning*. The MIT Press, 2004.

[4] E. Anderssen, K. Dyrstad, F. Westad, and H. Martens. Reducing Over-Optimism in Variable Selection by Cross-Model Validation. *Chemometrics and Intelligent Laboratory Systems*, 84(1):69–74, 2006.

[5] B.R.T. Arnold, Van Deursen A., and Res Martijn. An Algebraic Specification of a Language for Describing Financial Products. In *ICSE-17 Workshop on Formal Methods Application in Software Engineering*, pages 6–13, 1995.

[6] T. Bäck and H.P. Schwefel. An Overview of Evolutionary Algorithms for Parameter Optimisation. *Evolutionary Computation*, 1(1):1–23, 1993.

[7] J. Baillie. URBI-Workshop-28March, 2006. URL `http://www.docstoc.com/docs/127294969/URBI-Workshop-28March`.

[8] J.C. Baillie. URBI: A Universal Language for Robotic Control. *International journal of Humanoid Robotics*, 2004.

[9] J.C. Baillie. Universal Programming Interfaces for Robotic Devices. In *sOc-EUSAI '05: Proc of the 2005 Joint Conference on Smart Objects and Ambient Intelligence*, pages 75–80. ACM, 2005.

[10] J.C. Baillie. Urbi: Towards a universal robotic low-level programming language. In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 820–825. IEEE, 2005.

[11] Betfair. Betfair Charges, 2013. URL `http://www.betfair.com/www/GBR/en/aboutUs/Betfair.Charges`.

[12] H.G. Beyer and H.P. Schwefel. Evolution Strategies – A Comprehensive Introduction. *Natural computing*, 1(1):3–52, 2002.

[13] H.G. Beyer and B. Sendhoff. Covariance Matrix Adaptation Revisited–the CMSA Evolution Strategy–. In *Parallel Problem Solving from Nature–PPSN X*, pages 123–132. Springer, 2008.

[14] J. Bochow, P. Raupach, and M. Wahrenburg. What do Market Makers Achieve? In *Surveys in Experimental Economics*, pages 229–249. Springer, 2002.

[15] R. Bouckaert and E. Frank. Evaluating the Replicability of Significance Tests for Comparing Learning Algorithms. In *Advances in Knowledge Discovery and Data Mining*, pages 3–12. Springer, 2004.

[16] A. Brown. Evidence of In-Play Insider Trading on a UK Betting Exchange. *Applied Economics*, 44(9):1169–1175, 2012.

[17] K. Busche and W.D. Walls. Decision costs and betting market efficiency. *Rationality and Society*, 12(4):477–492, 2000.

[18] L. Cao and C. Zhang. F-trade: an Agent-mining Symbiont for Financial Services. In *AAMAS '07: Proc. 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1–2, 2007.

[19] L. Cao, C. Zhang, and R. Dai. Organization-Oriented Analysis of Open Complex Agent Systems. *International Journal of Intelligent Control and Systems*, 10(2):114–122, 2005.

[20] L. Cao et al. Agent Services-Based Infrastructure for Online Assessment of Trading Strategies. In *IAT '04: In Proc. of the Intelligent Agent Technology, IEEE/WIC/ACM International Conference*, pages 345–348, 2004.

[21] J. Carlson and B. Lisper. An Interval-Based Algebra for Restricted Event Detection. In *Formal Modeling and Analysis of Timed Systems*, pages 121–133. Springer, 2004.

[22] J. Carlson and B. Lisper. An Event Detection Algebra for Reactive Systems. In *Proc. 4th ACM International Conference on Embedded Software*, pages 147–154. ACM, 2004.

[23] D. L. Cassidy. *It's when you sell that counts.* Orient Paperbacks, 2006.

[24] G.C. Cawley and N.L. Talbot. Efficient Leave-One-Out Cross-Validation of Kernel Fisher Discriminant Classifiers. *Pattern Recognition*, 36(11):2585–2592, 2003.

[25] H. Chafi, Z. DeVito, et al. Language Virtualization for Heterogeneous Parallel computing. *SIGPLAN Not.*, 45(10):835–847, October 2010.

[26] S. Chakravarthy and Q.C. Jiang. *Stream Data Processing: a Quality of Service Perspective: Modeling, Scheduling, Load Shedding, and Complex Event Processing*, volume 36. Springer, 2009.

[27] S. Chakravarthy and D. Mishra. Snoop: An Expressive Event Specification Language for Active Databases. *Data & Knowledge Engineering*, 14(1):1–26, 1994.

[28] S. Chakravarthy, V. Krishnaprasad, E. Anwar, and S.K. Kim. Composite Events for Active Databases: Semantics, Contexts and Detection. In *In Proc. of the International Conference on Very Large Data Bases*, pages 606–606, 1994.

[29] K.C. Chan. On the Contrarian Investment Strategy. *Journal of Business*, pages 147–163, 1988.

[30] P. Christoffersen. *Backtesting.* John Wiley & Sons, Ltd, 2010.

[31] K. Croxson and R.J. James. Information and Effciency: Goal Arrival in Soccer Betting. *The Economic Journal*, 2013.

[32] G. Cugola and A. Margara. Processing Flows of Information: From Data Stream to Complex Event Processing. *ACM Computing Surveys (CSUR)*, 44(3):15, 2012.

[33] F. Dabek, N. Zeldovich, et al. Event-Driven Programming for Robust Software. In *In Proc. of the 10th workshop on ACM SIGOPS European Workshop*, pages 186–189. ACM, 2002.

[34] A.V. Deursen. Domain-Specific Languages versus Object-Oriented Frameworks: A Financial Engineering Case Study. *Smalltalk and Java in Industry and Academia, STJA'97*, pages 35–39, 1997.

[35] M.J. Dixon and S.G. Coles. Modelling Association Football Scores and Inefficiencies in the Football Betting Market. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 46(2):265–280, 1997.

[36] D. Dreman. *Contrarian Investment Strategies : The Next Generation. Beat the Market by Going Against the Crowd.* Free Press, 1998.

[37] J. Dréo, A. Petrowski, P. Siarry, and E. Taillard. *Metaheuristics for Hard Optimisation.* Springer, 2006.

[38] M. Edward, S. Falcone, and R. Pedro. A Survey of Signature Based Methods for Financial Fraud Detection. *Computers & Security*, 28(6):381–394, 2009.

[39] O. Etzion and P. Niblett. *Event Processing in Action.* Manning Publications Co., 1st edition, 2010.

[40] E.F. Fama. Efficient capital markets: a review of theory and empirical work. *Journal of Finance*, 25:383–417, 1970.

[41] Professional Market Feeder. Laying on any selection whose price falls below a certan limit, 2013. URL `http://marketfeeder.co.uk/solutions/back-lay-dutching/lay-price-below-limit`.

[42] E.D. Feustel and G.S. Howard. *Conquering Risk: Attacking Las Vegas and Wall Street.* Academic Publishers, 2010.

[43] H. Föllmer, A. Schied, and T.J. Lyons. Stochastic Finance. An Introduction in Discrete Time. *The Mathematical Intelligencer*, 26(4):67–68, 2004.

[44] G. Ford. *Systems Trading for Spread Betting: An End-To-End Guide for Developing Spread Betting Systems.* Harriman House Limited, 2009.

[45] E. Franck, E. Verbeek, and S. Nüesch. Inter-Market Arbitrage in Betting. *Economica*, 80(318):300–325, 2013.

[46] G. Dumitrescu, X. Huang, W.J. Knottenbelt, D. Spanias, and J. Wozniak. Inferring the Score of a Tennis Match from In-play Betting Exchange Markets. In *Proc. 4th Mathematics in Sport Conference, Leuven, Belgium, June 2013 (to appear)*, 2013.

[47] L. Gagnon and G.A. Karolyi. Multi-Market Trading and Arbitrage. *Journal of Financial Economics*, 97(1):53–80, 2010.

[48] GamblingData. European Regulated Online Markets Data Report, 2012. URL `http://www.gamblingdata.com/files/EuropeanRegulatedMarketsJuly2012_0.pdf`.

[49] S. Gatziu and K.R. Dittrich. SAMOS: An Active, Object-Oriented Database System. *IEEE Quarterly Bulletin on Data Engineering*, 15:23–26, 1992.

[50] S. Gatziu and K.R. Dittrich. *Events in an Active Object-Oriented Database System.* Springer, 1994.

[51] S. Gatziu and K.R. Dittrich. Detecting Composite Events in Active Database Systems using Petri Nets. In *Research Issues in Data Engineering, 1994. Active Database Systems. Proc. 4th International Workshop on*, pages 2–9. IEEE, 1994.

[52] N.H. Gehani and H.V. Jagadish. Ode as an Active Database: Constraints and Triggers. In *VLDB '91: Proc. 17th International Conference on Very Large Data Bases*, pages 327–336. Morgan Kaufmann Publishers Inc., 1991.

[53] R. Gencay. Optimisation of Technical Trading Strategies and the Profitability in Security Markets. *Economics Letters*, 59(2):249–254, 1998.

[54] R. Gençay and M. Qi. Pricing and Hedging Derivative Securities with Neural Networks: Bayesian Regularization, Early Stopping, and Bagging. *Neural Networks, IEEE Transactions on*, 12(4):726–734, 2001.

[55] A. Geppert and D. Tombros. Event-based Distributed Workflow Execution with EVE. In *Middleware'98*, pages 427–442. Springer, 1998.

[56] A. Ghosal et al. Event-driven Programming with Logical Execution Times. In *7th International Workshop on Hybrid Systems: Computation and Control (HSCC)*, pages 357–371, 2004.

[57] N. Hansen. The CMA Evolution Strategy: A Comparing Review. In *Towards a New Evolutionary Computation*, volume 192 of *Studies in Fuzziness and Soft Computing*, pages 75–102. Springer Berlin Heidelberg, 2006.

[58] Nikolaus Hansen. A CMA-ES for Mixed-Integer Nonlinear Optimization. 2011.

[59] L. Harris. *Trading and Exchanges*. Oxford University Press, 2003.

[60] A. Hinze, K. Sachs, and A. Buchmann. Event-Based Applications and Enabling Technologies. In *Proceedings of the Third ACM International Conference on Distributed Event-Based Systems*, page 1. ACM, 2009.

[61] F. Hoffmeister and T. Bäck. Genetic Algorithms and Evolution Strategies: Similarities and Differences. In *Parallel Problem Solving from Nature*, volume 496 of *Lecture Notes in Computer Science*, pages 455–469. Springer Berlin Heidelberg, 1991.

[62] J. Houghton. *Winning on Betfair For Dummies*. John Wiley & Sons, 2006.

[63] B.A. Huberman. Asynchrony and Concurrency. In *Neural Computers*, pages 455–465. Springer, 1988.

[64] R. Insley, L. Mok, and T. Swartz. Issues related to Sports Gambling. *Australian & New Zealand Journal of Statistics*, 46(2):219–232, 2004.

[65] R.A. Jaeger. Market Liquidity: Don't Know What You've Got 'Til It's Gone. *Trading*, 2011(1):14–18, 2011.

[66] D.S. Johnson and L.A. McGeoch. The Traveling Salesman Problem: A Case Study in Local Optimisation. *Local Search in Combinatorial Optimisation*, pages 215–310, 1997.

[67] D.R. Jones, M. Schonlau, and W.J. Welch. Efficient Global Optimisation of Expensive Black-Box Functions. *Journal of Global Optimisation*, 13(4):455–492, 1998.

[68] S.P. Jones, J.M. Eber, and J. Seward. Composing Contracts: an Adventure in Financial Engineering (Functional Pearl). In *ICFP '00: In Proc. of the 5th ACM SIGPLAN International Conference on Functional Programming*, pages 280–292. ACM, 2000.

[69] R. Kohavi et al. A study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. In *International Joint Conference on Artificial Intelligence*, volume 14, pages 1137–1145. Lawrence Erlbaum Associates Ltd, 1995.

[70] R.H. Koning and B. van Velzen. Betting Exchanges: The Future of Sports Betting? *International Journal of Sport Finance*, 4(1):42–62, February 2009.

[71] J.W. Kruisselbrink. *Evolution strategies for robust optimization.* PhD thesis, Leiden University, 2012.

[72] D. Laffey. Entrepreneurship and Innovation in the UK Betting Industry: The Rise of Person-to- Person Betting. *European Management Journal*, 23(3):351–359, 2005.

[73] Z. Laliwala, R. Khosla, P. Majumdar, and S. Chaudhary. Semantic and Rules Based Event-Driven Dynamic Web Services Composition for Automation of Business Processes. In *Services Computing Workshops, 2006. SCW'06. IEEE*, pages 175–182. IEEE, 2006.

[74] S. Levitt. Why are Gambling Markets Organised so Differently from Financial Markets? *Economic Journal*, 114(495):223–246, 2004.

[75] D.F. Lieuwen, N.H. Gehani, and R.M. Arlein. The Ode Active Database: Trigger Semantics and Implementation. In *ICDE '96: Proc. 12th International Conference on Data Engineering*, pages 412–420. IEEE Computer Society, 1996.

[76] D. Luckham. *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Pearson, 2002.

[77] D. Luckham. *Event Processing for Business*. Wiley, 2011.

[78] D.C. Luckham and J. Vera. An Event-Based Architecture Definition Language. *IEEE Transactions on Software Engineering*, 21(9):717–734, 1995.

[79] S. Luckner and C. Weinhardt. Arbitrage Opportunities and Market-Making Traders in Prediction Markets. In *E-Commerce Technology and the 5th IEEE Conference on Enterprise Computing, E-Commerce and E-Services, 2008 10th IEEE Conference on*, pages 53–59. IEEE, 2008.

[80] S. Luke. Essentials of Metaheuristics, 2013. URL `http://cs.gmu.edu/~sean/book/metaheuristics/Essentials.pdf`.

[81] M.J. Maher. Modelling association football scores. *Statistica Neerlandica*, 36(3):109–118, 1982.

[82] W.S. Mallios. *Forecasting in Financial and Sports Gambling Markets: Adaptive Drift Modeling*. Wiley, 2010.

[83] R.N. Mantegna. Hierarchical structure in financial markets. *The European Physical Journal B - Condensed Matter and Complex Systems*, 11(1):193–197, 1999.

[84] W. May, J.J. Alferes, and R. Amador. Active Rules in the Semantic Web: Dealing with Language Heterogeneity. In *In Proc. International Conference on Rules and Rule Markup Languages for the Semantic Web*, pages 30–44. Springer, 2005.

[85] O. Michel, Y. Bourquin, and J.C. Baillie. RobotStadium: Online Humanoid Robot Soccer Simulation Competition. In *RoboCup*, pages 580–590, 2008.

[86] R. Munting. *An Economic and Social History of Gambling in Britain and the Usa.* St. Martin's Press, 1996.

[87] Damianou N., Dulay N., Lupu E., and Sloman M. The Ponder Policy Specification Language. In *Policies for Distributed Systems and Networks*, pages 18–38. Springer, 2001.

[88] R.K. Narang. *Inside the Black Box. The Simple Truth About Quantitative Trading.* Wiley Finance, 2009.

[89] P.K. Newton and K. Aslam. Monte Carlo Tennis: A Stochastic Markov Chain Model. *Journal of Quantitative Analysis in Sport*, 5(3):1–42, 2009.

[90] I.H. Osman and J.P. Kelly. *Meta-Heuristics: Theory and Applications.* Springer, 1996.

[91] I.H. Osman and G. Laporte. Metaheuristics: A Bibliography. *Annals of Operations Research*, 63(5):511–623, 1996.

[92] M. Pagano. Financial Markets and Growth: An Overview. *European Economic Review*, 37(2-3):613–622, April 1993.

[93] R. Pardo. *The Evaluation and Optimization of Trading Strategies.* Wiley, 2008.

[94] G. Ramazan. The Predictability of Security Returns with Simple Technical Trading Rules. *Journal of Empirical Finance*, 5(4):347–359, 1998.

[95] R.G. Regis and C.A. Shoemaker. Constrained Global Optimisation of Expensive Black Box Functions Using Radial Basis Functions. *Journal of Global Optimisation*, 31(1): 153–171, 2005.

[96] J.D. Rodriguez, A. Perez, and J.A. Lozano. Sensitivity Analysis of k-Fold Cross Validation in Prediction Error Estimation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(3):569–575, 2010.

[97] J.D. Rodriguez, A. Perez, and J.A. Lozano. Sensitivity Analysis of K-Fold Cross Validation in Prediction Error Estimation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(3):569–575, 2010.

[98] D.L. Ruhm. Distribution-Based Formulas are not Arbitrage Free. In *Proceedings of the Casualty Actuarial Society XC*, pages 97–129, 2003.

[99] F. Schlottmann and D. Seese. Modern Heuristics for Finance Problems: A Survey of Selected Methods and Applications. In *Handbook of Computational and Numerical Methods in Finance*, pages 331–359. Springer, 2004.

[100] David G. Schwartz. *Roll the Bones: The History of Gambling*. Gotham Books, 2006.

[101] B. Selman, H. Kautz, Bram B. Cohen, et al. Local Search Strategies for Satisfiability Testing. *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*, 26:521–532, 1993.

[102] W.F. Sharpe. The Sharpe Ratio. *Journal of Portfolio Management*, 21:49–49, 1994.

[103] P. Shen and R.M. Starr. Market-Makers' Supply and Pricing of Financial Market Liquidity. *Economics letters*, 76(1):53–58, 2002.

[104] K. Simons. Value At Risk: New Approaches to Risk Management. *New England Economic Review*, (Sep):3–13, 1996.

[105] M. Sipper. The Emergence of Cellular Computing. *Computer*, 32(7):18–26, 1999.

[106] M.A. Smith, D. Paton, and W.L. Vaughan. Market Efficiency in Person-to-Person Betting. *Economica*, 73(292):673–689, 2006.

[107] M. Stonebraker, U. Çetintemel, and S. Zdonik. The 8 Requirements of Real-Time Stream Processing. *ACM SIGMOD Record*, 34(4):42–47, 2005.

[108] Google Tech Talks. Urbi a new Parallel & Event-driven Script Language for Robotics, Games and more, 2008. URL `http://www.youtube.com/user/GoogleTechTalks#p/search/0/FFLiIu2PtHQ`.

[109] D. Terrell and A. Farmer. Optimal betting and efficiency in parimutuel betting markets with information costs. *The Economic Journal*, pages 846–868, 1996.

[110] R.H. Thaler and Ziemba. Anomalies: Parimutuel Betting Markets: Racetracks and Lotteries. *The Journal of Economic Perspectives*, 2(2):161–174, 1988.

[111] E. Tsang, J. Li, and J.M. Butler. EDDIE beats the bookies. *Software-Practice and Experience*, 28(10):1033–1044, 1998.

[112] E. Tsang, P. Yung, and J. Li. EDDIE-Automation, a Decision Support Tool for Financial Forecasting. *Decision Support Systems*, 37(4):559–565, 2004.

[113] E. Tsang, S. Markose, and E. Hakan. Chance Discovery In Stock Index Option And Futures Arbitrage. *New Mathematics and Natural Computation (NMNC)*, 1(03):435–447, 2005.

[114] C. Ururahy and N. Rodriguez. Alua: An Event-Driven Communication Mechanism for Parallel and Distributed Programming. *PDCS'99*, 1999.

[115] C. Ururahy, N. Rodriguez, and R. Ierusalimschy. ALua: Flexibility for Parallel Programming. *Computer Languages, Systems & Structures*, 28(2):155–180, 2002.

[116] A. van Deursen, P. Klint, and V. Joost. Domain-specific languages: An annotated bibliography. *SIGPLAN Not.*, 35(6):26–36, June 2000.

[117] A. Van Deursen, P. Klint, and J. Visser. Domain-Specific Languages: an Annotated Bibliography. *ACM Sigplan Notices*, 35(6):26–36, 2000.

[118] J. Vecer, T. Ichiba, and M. Laudanovic. Parallels Between Betting Contracts and Credit Derivatives: Lessons Learned from FIFA World Cup 2006 Betting Markets. *Science*, (2005):1–15, 2006.

[119] N. Vlastakis, G. Dotsis, and R.N. Markellos. How Efficient is the European Football Betting market? Evidence from Arbitrage and Trading Strategies. *Journal of Forecasting*, 28(5):426–444, 2009.

[120] F. Wang, S. Liu, and P. Liu. Complex RFID Event Processing. *The VLDB Journal*, 18 (4):913–931, 2009.

[121] J. Weinman. *Cloudonomics: The Business Value of Cloud Computing*. ITPro collection. Wiley, 2012.

[122] T. Weise. Global Optimisation Algorithms–Theory and Application. *Self-Published,*, 2009. URL `http://www.it-weise.de/projects/book.pdf`.

[123] Wikipedia. Betting Exchange — Wikipedia, The Free Encyclopedia, 2010. URL `http://en.wikipedia.org/wiki/Betting_exchange`.

[124] Wikipedia. Basis Trading — Wikipedia, The Free Encyclopedia, 2012. URL `http://en.wikipedia.org/wiki/Basis_trading`.

[125] Wikipedia. XML — Wikipedia, The Free Encyclopedia, 2014. URL `http://en.wikipedia.org/wiki/XML#Criticism`.

[126] L.V. Williams and D. Paton. Why are some favourite-longshot biases positive and others negative? *Applied Economics*, 30(11):1505–1510, 1998.

[127] D.H. Wolpert and W.G. Macready. No Free Lunch Theorems for Optimisation. *Evolutionary Computation, IEEE Transactions on*, 1(1):67–82, 1997.

[128] J. Xingyi, L. Xiaodong, K. Ning, and Y. Baoping. Efficient Complex Event Processing over RFID Data Stream. In *Computer and Information Science, 2008. ICIS 08. 7th IEEE/ACIS International Conference on*, pages 75–81. IEEE, 2008.