## Algorithmic Trading Application – v0.1

# Idea I:

## Overview:

This application will obtain stock & option data for stated symbols. Then using various predictive methods and the constants associated with them it will:

1. Collect data until the constants effecting the algorithm reach a suitable standard deviation (SD).
2. This means the constants used in these algorithms will be iteratively calculated using realtime data over the period of up to a week until the set standard deviation is met.
3. Once this SD is reached the algorithm will use the constants as iteratively determined and commence simulated trades.
4. If the simulated trades maintain a sustainable profit for a specified amount of time the algorithm will begin live trading.
5. If a stop-loss value is hit the algorithm will cease trading for that particular symbol.

## Architecture:

1. WPF
2. Windows Azure (eventually)
3. Domain Driven Design
4. ORM – Dapper – Micro-ORM – High performance
5. Munq IOC
6. Big Data approach, store as much data locally as possible.

## Initial Steps:

1. WPF Form containing autocomplete search for companies.
2. On select of company the company is persisted in list (checkbox).
3. A graph is populated with realtime data for each the companies.
4. This data can be manipulated all the properties of the data can muted.
5. Another page for the analysis of data where per symbol calculated graphs are produced. (use quantlib to generate this)
6. Database contains symbol and order details.

## Predictive Algorithms:

### I:

1. Using a chunk of realtime data by time interval for instance. It quantifies the change between each tick in the data.
2. It then looks at historical data and checks to see if there are any matches to the pattern of data and from this data is gives an indicator to whether the change is likely to be positive or negative and for how many ticks this is likely (this is used to purchase stock and then sell).

### II: (used with I to confirm market direction)

1. Searches various news feeds via API’s and obtains data about a company/symbol.
2. The data is discretized for hourly, daily and weekly intervals.
3. Using the news/data about the company, various statistical methods are applied to see whether the news is generally positive or negative.

## Development:

### Sprint I:

1. Realtime model of data for a symbol.
2. Input -> Symbol
3. Output -> Dynamic model that contains all market information for that particular symbol.

### Sprint 2:

1. Database to store collected realtime data for up to one week for each symbol.

### Sprint 3:

1. WPF UI for data visualisation.
2. Main page:
   * Autocomplete for companies
   * Listview for companies being viewed, with checkbox to toggle displayed data which is realtime.
   * Toggles for displaying various pieces of market information, default is just realtime stock price (could be volume etc).
   * Toggles to show analysis or particular information (i.e. extrapolated curve that is being generated by predictive algoithms).

### Sprint 4:

1. Design predictive algorithms.
   * Store constants in db under algorithm name.
   * Implement algorithm to take in realtime market data and return indicator could be % likely to increase/decrease.

# Idea II:

## Overview:

This application will utilise machine learning to study historical market data to determine optimal future conditions.

Example Scenario:

Firstly historical market data of a large standard deviation will need to be obtained in order to maximise the number of observed patterns of the security.

1. Perhaps a highly volatile stock over a number of years, including recession trends.

Then per specified interval the derivatives of price with respect to the time interval (1 day, per tick, etc.) will be calculated.

Initially these derivatives will give

1. The velocity of the price (1st order differential, either negative price change or positive)
2. The change in market price at over DT (delta time, 2nd order).
3. The change in the rate of change (3rd order).

These derivatives can then be used to identify any common patterns i.e. the price of a stock may start to decline after a certain peak acceleration in price. This can be evaluated by analysing historical price peaks and indicators surrounding the peak i.e. maximum velocity, acceleration and the change in acceleration around each peak.

## Architecture:

* ASP.NET Restful Web API Angular JS, Bootstrap see Quandl (<http://www.quandl.com/stocks/goog>)
* Munq IoC
* Dapper ORM

## Design:

1. Create SSDT package to connect to web and pull down CSVs. Store data as flat table perhaps table partitioning per symbol?
2. F# script will:
   1. Identify price peaks
   2. Calculate the differentials at the smallest time interval.
      1. Repeat for the entire dataset.
      2. Store in separate table
   3. This data will then be analysed for repeating patterns.
3. Use analysis data to match against future data for the symbol. I.e. creation algorithm, perform analysis. Then collect a new dataset for that stock and then see if trends can be identified.