Developing an Automated Trading API

Madeline F. Briere & John Board, Ph.D.

Duke University | Pratt School of Engineering

Fitch & Co.

Abstract

This project centers on the development of an *Application Programming Interface*[1] for the automated trading of positions in the stock market. An external strategy component (not discussed here) is used to generate a map of trading instructions, which serves as input for the API. The Interactive Brokers *Trader Workstation*[2] is then used to submit the corresponding requests. The final product is capable of queuing jobs, placing orders, and requesting market data. It utilizes extensive error handling, resulting in a robust and effective application.

Background

- *Algorithmic Trading* is a method of executing large batches of automated and pre-programmed trading instructions[3]
  - Using intelligent strategies and drawing on existing market variables, one can algorithmically predict the instructions that will yield the highest financial returns
- An *Application Programming Interface* (API) is used for communication between various software components[1]
  - In this project, these components are the Interactive Brokers *Trader Workstation* (TWS) and a strategy component (not described here)
  - TWS submits real-world trading instructions
  - The API must submit requests to TWS and handle the responses of these requests
  - The API must also handle erroneous requests and the resulting application errors

Goals

Project Goals:

To develop an API that will:

- Interface between a trading strategy and a trading application
- Complete requests and return current market data
- Be robust and capable of handling failures (i.e., failure to complete orders, invalidated data, application failure)

Team Goals:

To produce a flexible platform:

- For automated trading of stock market positions
- Using an interchangeable strategy
- Able to run using both paper trading (for testing) and live trading (for actual trials)

API Requirements

	High-Level
Submit TWS requests	(1) Place orders to BUY/SELL/SHORT positions, (2) Request current market data (prices, daily returns)
Receive TWS responses	(1) Record the completion of jobs, (2) Receive and store responses to requests
Interface with Strategy	Translate mapping of portions/positions to orders through the TWS application
	Low-Level
Handle failures	(1) Consistently: catching identical failures in same way, (2) Robustly: catching all potential failures, (3) Verbosely: returning information regarding the failure to the user
Maintain job order	Submit requests in the order they are placed to maintain sequential logic
Parallelize tasks**	Complete independent tasks in parallel for efficiency

Figure 1: Requirements for the desired API, which communicates between TWS and a flexible strategy component

***Parallelization is more relevant to efficiency than functionality (the focus of this stage)*

Final Product

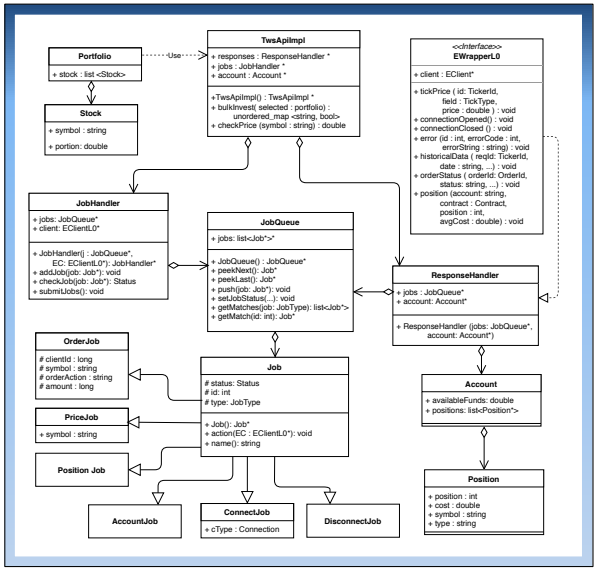

Figure 2: UML Diagram of the final API product, excluding external/imported resources such as the IB API [4]

- The final API layout can be seen in *Figure 2*
  - ***JobHandler****:* Sends *Jobs* to TWS via *JobQueue*
  - ***ResponseHandler***: Handles responses from TWS
  - ***Account***: Stores account data (positions, available funds, etc.)
  - ***TwsApiImpl*:** Accepts a strategy-based *Portfolio* and updates the connected account to reflect the desired portfolio
- The final project satisfies all objectives shown in *Figure 1* (except for parallelization)
- Final distributed work flow shown in *Figure 3*

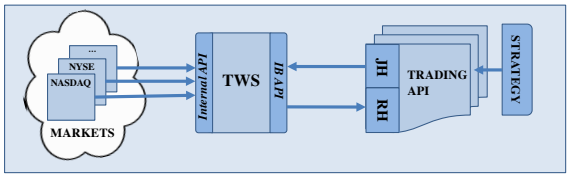

Figure 3: Distributed work flow: (1) Stock markets, (2) TWS, (3) (Multiple instances of) our trading application (where JH is the JobHandler and RH is the ResponseHandler)

Discussion

The final product is capable of consistent order placement and market data retrieval. It satisfies the majority of the initial requirements, leaving only parallelization left to complete. This will be more relevant when we consider efficiency, rather than functionality. In the upcoming months, the functionality of the API will be expanded to accommodate queries from the strategy portion of the product. Further error-catching will be added to protect against the placement of erroneous transactions. The end goal is to switch from paper trading to live trading and see actual returns.

Acknowledgements

Intellectual property of *Fitch & Co*.
Completed as Independent Study for *Duke University*

References

1. Elsevier. "What's an API? 5 Things You Need to Know to Stay Current." *Elsevier Connect*, www.elsevier.com/connect/whats-an-api-5-things-you-need-to-know-to-stay-current.
2. "Integrated Investment Management." *Low-Cost Online Trading | Interactive Brokers*, www.interactivebrokers.com/en/home.php.
3. "Definition of 'Algo or Algorithmic Trading' - NASDAQ Financial Glossary." *NASDAQ.com*, www.nasdaq.com/investing/glossary/a/algo-trading.
4. "IB API." *IB API | Interactive Brokers*, www.interactivebrokers.com/en/index.php?f=5041.