Reference test harness for algorithmic trading platforms

Victoria Leonchik
Exactpro Systems
victoria.leonchik@exactprosystems.com

Alexey Sukhov
Exactpro Systems
alexey.sukhov@exactprosystems.com

Eugene Ushakov
Exactpro Systems
eugene.ushakov@exactprosystems.com

Iosif Itkin
Exactpro, LSEG
iosif.itkin@exactpro.com

Anna-Maria Lukina
Exactpro Systems
anmay@exactpro.com

***Abstract*—The safety and stability of algorithmic trading software is an ongoing concern for exchanges, market participants and the society in general. Financial regulators worldwide are trying to create effective rules to prevent self-enforced market volatility and technology crashes caused by computer-aided trading. Specifying relevant requirements for dynamic software verification of algorithmic trading platforms remains is an on-going task. Yet, there has been little progress to date in locating efficient and commonly accepted approaches. This paper introduces a reference test harness implementation for algo trading platforms created by the authors.**

***Keywords*— trading, algorithm, strategy, testing.**

I. Introduction

The days of open outcry trading and trading pits are almost gone replaced by a new type of trading - algorithmic or electronic trading. Algorithmic trading is the process of using computers which execute a defined set of instructions to place orders to generate profits with speed and frequency impossible for a human to achieve [1].

Algorithmic trading became very popular during the last decade - about 40% of all financial operations are based on algorithms. Algo-trading by its nature is black box trading and there are still lots of concerns and question marks around this topic.

Algorithmic trading has received substantial attention from the society following high profile events, such as Flash Crash in 2010 when American indices (S&P 500, Dow Jones and Nasdaq 100) collapsed and recovered very rapidly [2,3], and the Knight Capital runaway algo disaster in 2012 when the firm lost $450 million in 45 minutes [4,5,6].

Regulators have started to recognize the value of algorithmic trading in the market place, but there is still concern about its safety and market risks caused by rogue algorithms. Regulators are trying to create rules and obligations regarding algorithm building, testing and deployment which could help prevent financial disasters and ensure that trading algorithms are safe and reliable.

The next section of this paper describes the role of algorithmic trading and associated legal framework. The third section introduces a reference test harness for algotrading platforms created by the authors. Section four drills into auxiliary test algo types targeted at modelling realistic market microstructure. The last section describes metrics and characteristics measured during algo trading platforms testing.

II. The role of Algorithmic Trading for Financial Market Rules and Regulations

The SEC (The United States Securities and Exchange Commission) started to formulate new rules back in 2013 and more recently FINRA published a list of suggested effective practices for firms engaging in algorithmic strategies [7]. An essential component of effective policies and procedures is testing of algorithmic strategies prior to launching them in Production.

Furthermore, the Hong Kong regulator (SFC - Securities & Futures Commission) has also worked on creating efficient rules which could be applied during the algorithm certification process: “A licensed or registered person should ensure that the algorithmic trading system and trading algorithms it uses or provides to clients for use are adequately tested to ensure that they operate as designed” [8].

ESMA has published a discussion paper for MiFID II / MiFIR which will be effective from January 2017. It includes a detailed description of the algorithmic testing procedure: “An investment firm that engages in algorithmic trading shall have in place effective systems and risk controls suitable for

the business it operates to ensure that its trading systems are resilient and have sufficient capacity" [9].

The main requirements of the SEC and ESMA can be found in the table below. (Fig.1)

SEC	ESMA
• Conducting testing: confirmation that core code components operate as intended and do not produce unintended consequences	• Clearly delineated development and testing methodologies
• Quality Assurance process should be separated from any development work	• Testing methodologies should include performance simulations / back-testing and non-live testing within a trading venue testing environment
• Periodically evaluating test controls	• Ensure that tests are commensurate with the risks that this strategy may pose to itself and to the fair and orderly functioning of the markets operated by the trading venue
• Data integrity, accuracy and workflow validation	• Periodically evaluating test controls
• Recording of all testing protocols and results • Conducting all testing in a development environment that is segregated from production	• Investment firms should ensure that the production and testing environments are kept segregated at all times

Fig.*1* SEC and ESMA requirements

One of the essential parts of algorithm testing, based on regulator requirements, is specific testing in a non-live trading environment. This would allow those involved in the certification process to make a correct assessment of algorithm risk, profitability and efficiency.

III. Reference Test Harness

We would like to introduce our view on testing solutions which could be used for testing of different algorithms and could be easily adapted to any trading platform required. The reference test harness consists of the following components:

- Algorithmic trading platform under test;
- One or several matching engines acting as execution venue simulators;
- Competing test algorithms to simulate market impact;
- Passive testing tools to gether quality, performance and efficiency stats for the algo systems and strategies under test;
- Customized order entry and market data gateways;
- Market surveillance system;
- Auxiliary proxies to control test execution.

At the core of our solution we are using an exchange matching engine as a market simulator. The authors have tried several exchange systems, including those developed by LSEG Technology services divison companies - MillenniumIT and GATElab. Let consider a full multi asset class matching platform developed by GATElab – Exchangepath – 100µs.

It is a matching engine with proven efficiency and can be used as a full replacement for a live trading system engine. The major advantages of this matching engine are:

- 100,000+ transactions per second;
- 50,000+ market data notifications per second;
- Low start up and running costs;
- Low latency – less than 100µs at the end-user gateway [10].

These conditions give us an opportunity to place a trading algorithm under test in a Production-like environment and get results which are closer aligned to what we would expect in a Production environment. Trading algorithm under test can submit orders into the matching engine and receive back execution reports and related market data. It is possible to deploy several matching engines to simulate multiple markets.

To decrease test harness hardware footprint, the authors have introduced a replay of historical data known as backtesting. Backtesting is useful as it could demonstrate the efficiency of an algorithm from a historical point of view. On the other hand, quite often backtesting tells you very little about future profitability. Because of this, backtesting is both a blessing and a curse. Many portfolio managers use backtesting to prove that a strategy is viable, but they fail to evaluate a number of issues that might be missed during backtesting as general market sentiment cannot be predicted.

It is not possible to achieve 100% accuracy in the trading day replace when dealing with highload distributed systems [11]. As soon as a client algorithm is connected to the system it will have an impact on historical data and change it, which may lead to incorrect results in the evaluation of algorithm risk and productivity. That is why a trading firm should be concerned about returning historical data to its original state without proper compensation of strategy impact. Backtesting can produce positive results which can be too far from results achievable in real conditions. In other words, during algorithm testing we should take into account the impact of the algorithm on historical market data replay. Thus for better testing this impact should be suppressed - market data should be restored by using counter flow models.

A testing tool should create counter-flows in response to user-generated, non-historic submissions. The goal of any counter-flow model is to replicate sufficiently the reaction of the market on a user strategy through the generation of additional flow of events which would be united with modified historical and user-generated events. A counter-flow model has very complex logic and consumes a lot of hardware resources that leads to increased costs and testing time.

As an alternative to using the counter-flow model and overall testing process improvement we suggest to add four points in to the system which would be configurable for any system and could help compensate testing strategy impact (fig.2):

- Customized trading GWs;
- Customized Market Data GW;
- Latency and counterflow proxies;
- Simulated traders (Active Testing Tool):
  - *Arbitraging Minirobot*
  - *Minirobots emulating 'Slicing' algorithms*
  - *Minirobots emulating 'Synthetic' algorithms*
  - *Exchange simulated orders*
  - *Aggressive buyer/seller (Market panic scenarios)*
  - *"Bandit"-algos*

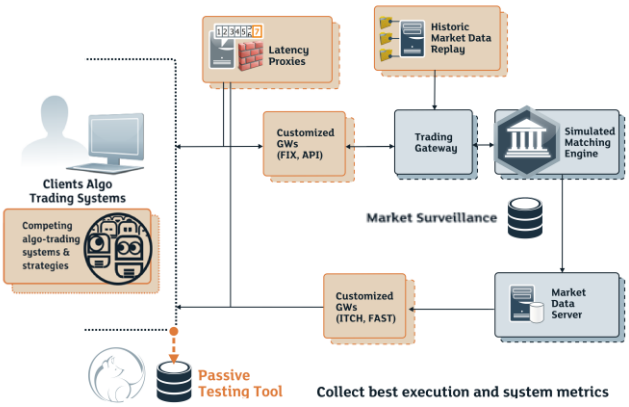

Fig.2 Test Harness for Algo trading system

Customized Trading Gateways and Market Data Gateway

In order to have an ability to put a client's algorithm in a disadvantaged position compared to another one which would be "faster", we can use customized Trading gateways. This would allow us to test the algorithm under different co-location conditions as we are able to control the latency of all messages sent or received by the client's algorithm.

Testing will answer the following question: how profitable an algorithm could be if outgoing/incoming messages have an acceptable time delay or what would happen to algorithm efficiency in case of increased latency?

The same customization could be applied to Market data gateways such as ITCH and FAST. It would give us an opportunity to evaluate the impact of different latencies in receiving market data messages on the performance of the trading algorithm.

Customized Trading gateways also allow us to change tags in sent/received client's messages. In this case we will be able to adjust our system to different clients who are using different sets of tags.

Surveillance system

In addition to testing efficiency of the trading algorithm we should take into account the involvement of control functions (such as Suveillance, Legal, Compliance, Controllers, Operations, etc.) as well as business objectives. These control measures must prevent disruptions to the fair and orderly functioning of the financial markets. Disruptions and incidents may cause damage for trader's activity, but potentially also for others traders too.

Despite the fact that an algorithm can show quite impressive results and make good profit, it does not mean that it can be used carelessly. Thus it makes sence to test the legality of algos. It may be worth to run it through Surveillance systems to make sure that trading firm will not have any issues with regulators in terms of basic market rules and the trading algorithm will not be considered abusive or disruptive [12].

Passive Testing Tool

While using highly complex algorithms it is important to store all the inbound and outbound data about all executed financial transactions and verify this data with the data of the client and the data in post trade. [13]

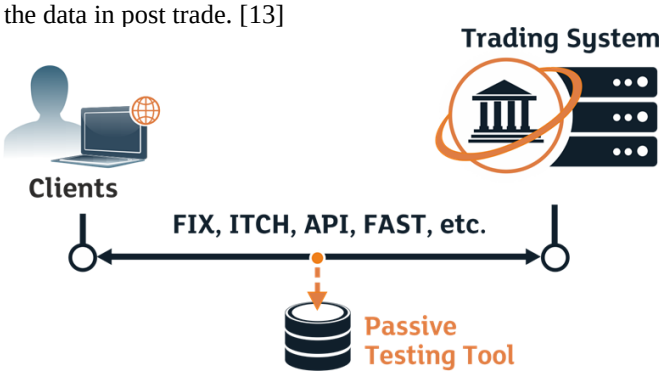

Fig.3 Passive Testing Tools in Trading Systems

Passive testing tools (Fig.3) are used for automated log collection, data structuring, monitoring, system behavior analysis and user certification [14]. Test tools allow analyzing high volume of data promptly, reacting to deviations in the system's behavior from requirements, and troubleshooting. (Fig. 4)

Item	Description
Testing Type	Passive Real-Time/Batch
Target SUT	Trading Platforms, Market Data Delivery and Post-Trade Systems
SUT Interface	Back-end (typically connected to message gateways / APIs, and DBs); GUI Testing Capabilities not supported
SUT Interaction Method	Inputs and outputs monitored by means of message capture and log parsing to analyze client activity and forecast system response; DB queries for data verification; files transfer, upload, export and comparison. Captured messages can be viewed and analyzed in real-time or post-factum

Protocols	Extant plug-ins for Industry-standard (FIX and dialects, FAST, SWIFT, ITCH, HTTP, SOAP, etc.) and Proprietary (MIT, SAIL, HSVF, RTF, RV, Reuters, Fidessa OA, Quant House, etc.) protocols. New plug-ins for additional protocols developed by request (codecs are shared between Sailfish and Shsha)
Test Scripts	Certification tests and data reconciliation may be performed by using ordinary SQL queries. Test message traffic generated in real-time or replayed from log files by other tool (e.g., Sailfish)
Test Management, Execution and Reporting	Integrated (Desktop front-end), allows for multiple simultaneous heterogeneous connections, consecutive execution of multiple planned scripts, test results summary and detailed test reports. Optional Big Button framework supported
Platform requirements	Low footprint cross-platform application, MySQL

Fig.4 Passive testing tool specification

Minirobots

Returning market data replay to its initial state can be achieved by introducing an arbitraging script formalized in one of the Minirobots tool which is to be used together with a tested algorithm in the same framework.

The Minirobots tool [10B], developed with the idea of simulating real traders' behavior in mind, is able to make decisions under specific market conditions in a common fashion, but at the same time has a certain degree of autonomy. Depending on what the testing needs are, each of the robots can act independently or jointly executing a particular trading strategy or simply replaying a stored list of orders. (Fig. 5)

Item	Description
Capacity & Precision	Hundreds – thousands of messages depending on the algorithm complexity. Millisecond precision
Testing Type	Active Multi-Participants (applicable for testing at the confluence of functional and non-functional testing)
Target SUT	Trading Platforms and Market Data Delivery Systems
SUT Interface	Back-end (typically connected to message gateways / APIs); GUI Testing Capabilities not supported
SUT Interaction Method	Message injection and capture to emulate multiple participants' activity in electronic markets (essential when there is a need to reproduce complex scenarios that can be created by trading algorithms)
Protocols	Extant plug-ins for Industry-standard (FIX and dialects, etc.) and proprietary protocols. New plug-ins for additional protocols developed by request
Test Scripts	Multi-threaded Java code specifying different liquidity profiles
Test Management, Execution and Reporting	Integrated (Web front-end), allows for multiple simultaneous heterogeneous connections, concurrent emulation of multiple participants, detailed test reports. Optional Big Button framework supported
Platform requirements	Written in Java

Fig.5 Minirobots specification

IV. Algo Test Agents Used to Simulate Liquidity and Market Impact

a. Arbitraging and Market Making Minirobots

Once arbitraging Minirobots has been implemented in the test environment, it starts sending contra-orders to add liquidity on a particular price level as soon as the order sent from the tested algorithm hits the 'market' bid or an offer. That is how replayed market data is restored to the realistic state, thereby reducing the potential impact of the tested algorithm on the event sequence of the analysed trading day. Arbitrage Minirobots are helpful when testing fragmented markets. They can move liquidity across simulated venues.

b. Minirobots emulating 'Slicing' algorithms

Another productive option is to deploy Minirobots tool emulating the behavior of market operator who is using 'Slicing' trading algorithms being widely spread at present among a variety of institutional investment funds. According to U.S. Commodity Futures Trading Commission [15] estimates, more than 90% of institutional traders use trading algorithms or other automated strategies to seek best execution for their clients. This is due to the fact that 'slicing' allows hugely sized trading orders to be executed by dividing them into small portions, which are then traded separately at different time during a trading day. This minimises the risk of orders being discovered and 'front-run' by competitive or 'dirty' traders.

The 'slicing' idea is most oftenly materialised in the usage of the so called 'Time Sliced' trading strategy. The Time Sliced algo splits an order into equal sized slices (an element of randomisation can be added too). The number of slices is determined by the start and end time and the duration between slices. Each slice is released to the market at intervals dictated by the duration between slices.

'VWAP' represents a more sophisticated implementation of a 'slicing' strategy. This algorithm is intended to manage the execution of an order in such way that it achieves the Volume Weighted Average Price (VWAP) for the order's instrument in the time period the trader has selected.

At a high level, the algorithm achieves this by releasing slices of the order at varying rates according to the analysis of historical market data. It uses additional execution logic to determine the pricing and expiry of each 'slice' created by the model.

The fact that Time Sliced/VWAP based order execution models is being popular in modern trading determines a necessity to use Minirobots tool simulating Time Sliced/VWAP trader together with a client's algorithm to understand its efficiency and competitive performance when trading is driven by someone else's automatic solution. [16,17]

c. Minirobots emulating 'Synthetic' algorithms

When it comes to better execution one can either use an automated solutions allowing the orders to be placed at the top of the market's queue. For instance, this is often a matter of immediate interest when there is a need to execute a trade at the very start of a trading day. Such algos, usually called 'Hammer', are attempting to send the order just before market open in order to reach the Exchange at the exact open time rather than wait to receive the market open signal. This is achieved by sending the order a few seconds before market open (where it will be rejected) and by continuously re-sending it until it is accepted by the Exchange. 'Hammer' algorithms behavior can also be mimicked by the Minirobots tool to see how the automated strategy tested interacts with it at the peak hours of a trading day.

If trading is circumstantial and the trader needs to place an order as a result of a particular price movement in an correlated instrument (or a group of correlated ones), another algo is used to release orders once the conditions pre-defined by a trader's logic are reached. This behavior simulated by the Minirobot tool would give a singular advantage where one wants to test 'pairs trading' or arbitraging algorithms.

d. Exchange simulated orders

Where a market data replay is recorded from the exchange that due to external reasons does not support natively some of the order types, there is always a possibility that stop, trailing stop, iceberg, ghost or market-if-touched orders are being simulated by someone else's automated solution. This in fact can affect the test results of a chosen algorithm and thus the Minirobot tool sending 'exchange simulated' orders might be another example of a more sophisticated testing approach.

e. Aggressive buyer/seller (Market panic scenarios)

Moreover, Minirobots who act like an aggressive buyer or seller can be deployed to the testing environment. With such a script implemented, it will be possible to re-create conditions of the so called 'panic' buying or selling in a particular derivative contract, underlying instrument, or a market composite index. Beyond all doubt market 'panic' is a once-off experience which may stimulate irrational price movements frequently spurring algo-traders to react irrationally and wildly. Such events a the well-known 'Flash Crash' are subject of anxiety due to a potential devastating impact on the financial stability of the stock or even the economy. Emulating this by using the Minirobots tool provides a benefit to assure that tested algorithm would not fail under stress market events. [18,19]

f. 'Bandit'-algos

With algo trading increasing in popularity in the financial markets there is a worrisome trend for anti-HFT or so called "Predatory algorithms" to be used by traders whose aim is to manipulate stock prices, forcing others to react to their benefit. Despite the MiFID I and Dodd-Frank legislative acts going into effect as long as 5 years ago, these illegal trading practices are still popular due to the fact that they are highly profitable and not always easy to detect, unfortunately. All these require us to introduce a 'bandit script` - having a 'predatory' logic - into the environment to see how it will affect the tested algorithm in terms of its performance characteristics.

Based on 'Proposed Guidance on Certain Manipulative and Deceptive Trading Practices' issued by Investment Industry Regulatory Organization of Canada (IIROC) [20,21] we might use the Minirobot tool to re-create the following abusive market behavior -

1) 'Layering'

A strategy which initiates a series of orders and trades (sometimes along with spreading false rumours in the marketplace) in an attempt to ignite a rapid price movement either up or down and induce others to trade at artificially high or low prices. An example is a "layering" strategy whereby a market participant places a bonafide order on one side of the market and simultaneously "layers" the book with non-bona fide orders on the other side of the market to bait other market participants into reacting to the non-bona fide orders and trade with the bonafide order.

2) 'Quote-stuffing'

The practice of placing an unusual number of buy or sell orders on a particular security and then immediately cancelling them to "flood" the trading systems with excessive market data messages. An objective may be to increase data latencies for marketplaces or other market participants in order to create "information arbitrage" opportunities.

3) 'Spoofing'

A practice when limit orders that are not intended to be executed are used to manipulate prices. Some strategies are related to the open or the close of regular market hours that involve distorting disseminated market imbalance indicators through the entry of non-bonafide orders, checking for the presence of an "iceberg" order, affecting a calculated opening price and/or an aggressive trading activity near the open or close for an improper purpose.

4) 'Abusive liquidity detection'

Large orders (disclosed or iceberg) are entered during the pre-open or employ "pinging" orders to detect the existence of a large buyer or seller with the intent to trade ahead of, rather than with, the large buyer or seller. After a profitable price movement, the trades are reversed, or if the price moves contrary to the position taken, the trading interest of the large buyer or seller may be viewed as a free option to trade against.

All in all, the Minirobots tool allows us to emulate any existing market participant behavior that can be used in client-to-algo or client-vs-algos mode within the same testing environment. This in fact offers a unique possibility to re-create Production-like conditions, and hence to assure that a tested algorithm is operating adequately with no real risk of losing money.

V. Algorithm efficiency and riskless criteria

The main purpose of trading algorithm testing is proving its efficiency and reliability. No one will use algorithms which only generate financial losses. There are many criteria which could be used for the evaluation of algo productivity.

Thus we decided to divide all criteria into two sections Technical and Business efficiency:

Technical Efficiency Criteria

This section is based on ISTQB classification [22].

Functional criteria

The ability of an algo to produce correct outputs for the inputs it receives according to specification. The less number of existing errors an algorithm contains, the lower the expectations of potential losses and fines are. The test harness developed by the authors parses log files to search for error and warning messages. It also analyses the consistency of the data collected by passive testing tools. Simulated matching engines and gateways reject incorrectly formatted messages and orders that fail to pass the risk controls.

Non-Functional criteria

1) Performance

How does a trading algorithm perform in terms of responsiveness and stability under a particular workloads (market data streams)? Answering this question helps Quality Assurance in understanding how an algo can cope with the number of data feeds in processes, the number of exchanges it trades on, and the types of securities it can trade. Data collected from the network capture is used to estimate internal latencies within the trading algo, including the time from market data updates to issuing orders into the market.

2) Scalability

This is the capability of an algorithm to continue functioning well under a growing amount of work, or its potential when the algo is provided with more resources (hardware mostly) in order to accommodate that growth or to meet a user need. The presence of the scalable matching engines as simulated markets allows running scalability tests for algorithmic trading platforms.

3) Reliability

The ease of an algo to perform its required functions well for a specified period of time under different specific test conditions or for a specified number of operations. Test harness automation enables us to repeat the tests many times to check the systems reliability.

4) Efficiency

The capability of an algorithm to provide appropriate performance under stated conditions, relative to the amount of resources used. Hardware metrics are collected from the trading platform to test its technical efficiency.

5) Maintainability

This is the ease with which an algorithm can be modified to correct defects, modified to meet new requirements (e.g. market conditions, regulatory acts changes), or modified to make future maintenance easier. The Knigh Capital case shows that the maintability and the ability to monitor the system is what prevents a problem from turning into a disaster.

6) Recoverability

This is the capability of an algorithm to re-establish a specified level of performance and recover the data directly affected in case of failure. Failover and recovery tests should be included into the systems testing scope.

Business Efficiency Criteria

Achieving the best possible trading price does not guarantee that the algorithm will make a profit. Best execution is a wider definition [23]. Depending on the underlying trade or investment idea on which a trading algorithm has been built, best execution includes taking all appropriate steps to achieve the best possible outcome along several dimensions. These dimensions are:

- Price (execution, price improvement, spread capture);
- Cost (explicit, market impact, adverse selection);
- Probability of execution;
- Liquidity and volatility.

There are a number of common execution performance criteria that the trading algorithm is characterized by. The most general and relevant criteria used to measure execution performance is Implementation Shortfall (IS). This approach has become an industry standard as it captures the difference between the price that an algo decided to trade and the final execution price (including commissions, taxes, etc.) for a trade. This is also known as “slippage”.

At the level of algorithmic strategy, best execution is achieved by balancing multiple conflicting goals such as best trade price, minimal market impact, optimal time and liquidity allocation, and highest possible completion rate [24].

Passive test tools implemented as part of the test harness allow capturing business efficiency parameters for every test execution and storing them into the database for regression analysis. It is necessary to repeat every test many times as real markets are not deterministic and good simulated markets are not deterministic either. Every particular test run will lead to a slightly different result. Queries executed against captured data give us the necessary analytics on the systems’ behavior and enable comparison between various versions of the systems under test.

VI. Conclusion

In this paper we have introduced our view on the development of a trading platform simulator which could be used for testing of trading algorithms and strategies. The main advantages of our approach are flexibility and simple tuning of simulator configuration depending on testing purposes.

We are planning to expand our system and develop an automated solution which could restart and change the simulator's parameters and, furthermore, collect and aggregate statistics and logs after each testing cycle. It would make testing more efficient, adjustable and understandable for clients.

We believe that the financial regulators will pay much more attention to algorithmic trades and the importance of testing trading algorithms in the future. Therefore, trading simulator solutions will become more and more popular and will be in higher demand as time goes on.

References

[1] Gov.UK, *Future of computer trading in financial markets: an international perspective,2012*, https://www.gov.uk/government/publications/future-of-computer-trading-in-financial-markets-an-international-perspective

[2] U.S. Commodity Futures Trading Commission, U.S. Securities & Exchange Commission, *Findings Regarding the Market Events of May 6, 2010*, Report of the Staffs of the CFTC and SEC to the Joint Advisory Committee on Emerging Regulatory Issues, https://www.sec.gov/news/studies/2010/marketevents-report.pdf

[3] E. Wes Bethel, D. Leinweber, O. Rübel, K. Wu, *Federal market information technology in the post flash crash era: roles for supercomputing*, WHPCF '11: Proceedings of the fourth workshop on High performance computational finance, 2011.

[4] SEC Release No. 70694, *In the Matter of Knight Capital Americas LLC*, 2013, https://www.sec.gov/litigation/admin/2013/34-70694.pdf

[5] A. Kriger, A. Pochukalina, V. Isaev, *Reconciliation Testing Aspects of Trading Systems Software Failures*. Preliminary Proceedings of the 8th Spring/Summer Young Researchers' Colloquium on Software Engineering (SYRCoSE 2014), ISBN 978-5-91474-020-4: 125 p. , 2014, http://syrcose.ispras.ru/2014/files/SYRCoSE2014_Proceedings.pdf

[6] G.Baxter, J. Cartlidge, *Flying by the seat of their pants: what can high frequency trading learn from aviation?* ATACCS '13: Proceedings of the 3rd International Conference on Application and Theory of Automation in Command and Control Systems, 2013.

[7] Regulatory Notice 15-09, *Equity Trading Initiatives:Supervision and Control Practices for Algorithmic Trading Strategies*, https://www.finra.org/sites/default/files/notice_doc_file_ref/Notice_Regulatory_15-09.pdf

[8] Securities and futures commission, *Code of conduct persons licensed by or registered with the securities and futures commission,2013*, http://en-rules.sfc.hk/net_file_store/new_rulebooks/h/k/HKSFC3527_1868_VER50.pdf 18.10

[9] The European Securities and Markets Authority, *MiFID/MIFIR Discussion Paper*, http://www.esma.europa.eu/system/files/2014-548_discussion_paper_mifid-mifir.pdf

[10] GATElab, *The Dive Deep into Liquidity Pools*, http://www.gatelab.com/products/matching.htm

[11]P. Protsenko, A. Khristenok, A. Lukina, A. Alexeenko, T. Pavlyuk, I. Itkin, *Trading Day Logs Replay Limitationas and Test Tools Applicability*, TMPA'2014: Proceedings of Annual International Workshop-Conference Tools & Methods of Program Analysis (TMPA), 2014

[12] U.S. Securities and Exchange Commission, *Regulatory Actions*, https://www.sec.gov/rules.shtml

[13] A.A. Averina, N.A. Antonov, I.L. Itkin, *Special features of testing tools applicable for use in trading systems production*, TMPA'13: Proceedings of Annual IEEE/ACM International Workshop-Conference Tools & Methods of Program Analysis (TMPA), 2013

[14]A. Alexeenko, A. Matveeva, D. Sharov, P. Protsenko, I. Itkin, *Compatibility Testing of Clients Protocol Connectivity to Exchange and Broker Systems*, TMPA'13: Proceedings of Annual IEEE/ACM International Workshop-Conference Tools & Methods of Program Analysis (TMPA), 2013

[15] U.S. Commodity futures trading commission,http://www.cftc.gov/index.htm

[16] A. Madhavan, *VWAP Strategies, 2002*, http://itg.com/news_events/papers/TP_Spring_2002_Madhavan.pdf

[17] J. Bialkowski, S. Darolles, G. Fol, *Improving VWAP strategies: A dynamical volume approach, 2006*, http://www.ir.canterbury.ac.nz/bitstream/10092/4534/1/12624569_VWAP_2310_2006_for_JBF.pdf

[18] O. Steinki, *Business School, Algorithmoc Trading*, http://evolutiq.com/wp-content/uploads/2014/03/Algo-Trading-Intro-2013-Steinki-Session-8.pdf

[19] Fidessa, *Advanced trading tools*, http://www.fidessa.com/products/sell-side-solutions/advanced-trading-tools

[20] Compliance: Theory and Practice in the Financial Services Industry,*Market Conduct Rules*,http://www.inhouselegal.com.au/Compliance_Course/lecture_4.htm

[21] IIROC Notice 12-0221, *Rules Notice, 2012*, http://www.iiroc.ca/Documents/2012/f62c746a-b5c9-448a-b57f-f1c04c88de14_en.pdf

[22] ISTQB Glossary, http://astqb.org/glossary/

[23] Trader Planet, Trading 101: The M&Ms for Successful Trading, http://www.traderplanet.com/tutorials/view/161657-trading-101-the-m-amp-ms-for-successful-trading/

[24] Trader Planet, *What Traders Need to Know: Best Execution*, http://www.traderplanet.com/articles/view/162821-what-traders-need-to-know-best-execution/