CppCon 2024 Poster Submission Form

Name: Yuan Yuan
Email: yyuan133@bloomberg.net
Bio: Yuan Yuan is a senior software engineer on the electronic trading team at Bloomberg. He has more than 10 years of experience developing financial software, with a specialization in algorithmic trading and pricing systems. He is particularly interested in building low-latency quant trading systems and reliable software. He holds a master's degree in computer science from Columbia University.

Title: Behavior-driven Tests for Microservices-based Algo Trading System
Summary:
Introduction: This poster will introduce a test framework we use at Bloomberg to enable behavior-based tests in natural language style for a microservices-based algo trading system that our engineering team is responsible for.

Relevance: For our electronic trading business, it is important to thoroughly test algo behaviors. Although microservice components can easily be unit tested, it is often non-trivial to automate the system-level tests with external dependencies. Controlling external dependencies are especially important for algo tests since certain algo behaviors require hard-to-produce market/timing conditions to trigger. Thus a test framework is proposed to address these pain points:

- **End-to-end**: The test framework validates the end (client) to end (broker/exchange) algo behavior.
- **Controlled external dependencies**: All external dependencies including market data, reference data, exchange/executions, time, etc. are mocked and controlled to simulate any triggering condition.
- **Fully automated**: Running tests are integrated into CI so algo behaviors are regressed for any dependent code change.
- **Natural language layer:** Tests are written in natural language so they are easy to read and write, even for non-technical staff.

Discussion:
The test framework includes two layers:

- **A controller layer:** This layer provides a set of APIs (interfaces) to interact with the microservice system. It sets up an internal test version of the system with a collection of components, a mocked message channel, and mocked dependencies.
- **A BDD layer:** Using cpp-cucumber (open source), this layer maps natural language to C++ APIs at run time, allowing for rapid test writing/modifying without having to recompile C++ code.

Completion Status: Work has been completed; system is in use.