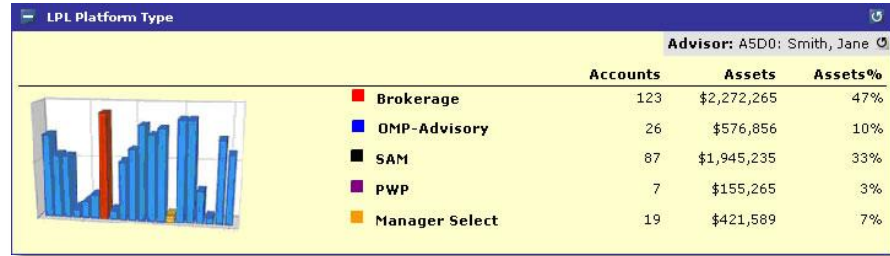


# LPL Financial Portfolio Manager

## Designing & Maintaining Mission-Critical Services

Independent advisors, banks, credit unions, and leading financial services companies depend on LPL’s state-of-the-art tools and technology to successfully manage their organizations. By providing independent investment research, dedicated service and training, and an open product architecture, LPL allows its customers to focus on giving sound investment advice to their clients.

Customers manage and run reports on portfolios for their clients using Portfolio Manager, which provides a suite of tools including Performance Engine, Pricing Service, and Reports Engine.



After proven successes on other projects, LPL enlisted Fairway to lead the Portfolio Manager team of 8 developers made up of internal LPL employees, on-site consultants, and remote consultants. Fairway’s leadership allowed LPL to:



- provide customers with accurate information anytime with a stable, maintainable product;
- improve modularity to respond quickly to customers’ feature requests; and
- increase productivity by applying best practices and a streamlined coding process.

Fairway was responsible for:

- gathering requirements,
- designing new components,
- supporting developers,
- managing team processes,
- implementing best coding practices,
- streamlining builds,
- setting the software configurations management processes, and
- taking part in inter-team design reviews.

When LPL hired Microsoft consultants to refactor critical code, Fairway drove the meetings to preserve LPL’s interests.

In an effort to improve team processes, Fairway set up a SharePoint site to transfer knowledge and created a process for team members to add information. The Wiki features greatly reduced dependence on email and became the main repository for all project information.

Fairway also implemented a new testing process to improve developer confidence and quicker turn-around before handing over code to the internal QA team. Team members implemented unit tests using Visual Studio Team System and tracked and assigned defects with the Test Director tool. As tech lead, Fairway required a thorough understanding of the

entire system to efficiently assign defects to the appropriate developers. While supporting the QA team, Fairway oversaw code complete and software builds and created the release schedule.

## Performance Engine

50% of all of LPL's online reports depend on a windows service called Performance Engine, which makes maintaining and enhancing it mission critical.

Unfortunately, this also meant that frequent, vital updates were often not given enough time to be implemented correctly, resulting in over 8000 lines of code in a single file and over 3000 lines in the largest function. With fourteen levels of nesting of varying depths in a single function, the service was regarded as the worst piece of software in the system. As a result, no one wanted to change it because no matter how small the change, seemingly unrelated features would break.

When the service started crashing periodically requiring a manual reboot, LPL made it a priority to redesign and re-architect it.

Fairway started by understanding the industry in order to refactor Performance Engine in a way that made sense and would be maintainable. The first step in the clean-up was to give meaningful names to all the variables. Next, Fairway created a large number of unit tests with known outcomes against which all modifications could be evaluated. Algorithms identified as broken were refactored to fix the errors. Additional defects were corrected, memory leaks were eliminated, and comments were added before the service was turned over to the internal testing team. Fairway also advised the testing team on how to use a manual tool to emulate various client users with differing permissions.

Throughout the process, Fairway documented the project on a Wiki. Visio flow-charts illustrated the updated architecture, each unit test was explained, and the new build and deploy process was described. This eased training and transition of responsibilities to LPL's internal resources.

Economy		
DJIA	10,799.23	-
NASDAQ	2,043.22	-
S&P	1,236.86	-
Funds-Test 5		
CHTAX	10.28	+0.58
CTBIX	9.49	-0.74
CHTMX	10.17	-
MOBIX	16.39	+0.61

## Pricing Service

### ROR Transaction History

Rights of Reinstatement Eligible Sell and Buy transactions for the (90 day) Fund Family Reinstatement Window are listed below.

Date	Buy/Sell	Symbol/ CUSIP	Share Class	Number of Shares	Price Per Share	Transaction Amount	Reinstates Into
08/24/2004	Sell	FRDX	CL-A	1142.875	\$17.50	\$20,000.00	CL-A
09/15/2004	Buy	FRMAX	CL-B	2142.875	\$14.00	\$30,000.00	
10/13/2004	Buy	FRDX	CL-A	500.000	\$20.00	\$10,000.00	
10/27/2004	Sell	FRMAX	CL-B	200.000	\$18.45	\$3,000.00	CL-C

With thousands of equities traded every day, managing the pricing data used to generate accurate reports is daunting. LPL's Pricing Service caches the data required by the reporting web page so that

users don't have to wait to get the data from the server every time they run a report. This also minimizes the database load as reports run. When the service started intermittently running out of memory and crashing the server, Fairway was needed to determine why the issue couldn't be reproduced in the development environment and ultimately fix the problem.

Performance counters native to the .NET runtime were set up on the production system to allow Fairway to monitor system resources. Fairway analyzed the network traffic results and correlated the data. The .NET heap usage during times of failure indicated that large objects were getting allocated but not freed. Looking through the code, Fairway found all the large objects and determined that the XML responses were the culprit. During periods of heavy network usage, incoming requests could not be serviced fast enough, which caused the

service to generate large XML streams that could not serve data back out fast enough and, thus, the service would run out of memory.

Once the problem was identified, Fairway recommended several solutions and evaluated them based on:

- the effort required,
- the long term impact to the systems,
- the effect on other systems, and
- the cost.

The least effort required would be to simply increase the system memory. Reducing the amount of data served by the application would solve the problem, but providing fewer days of prices would diminish the user experience by slowing down the reports. The best long term choice would be to stop using XML and convert to binary format; however, this would have required updating every client system, thus posing a risk to the stability of the system and consuming a lot of developer time. The compromise made based on budget, risk, and developer constraints was to isolate the changes to a single system and optimize the code to use less memory.

Optimizing the code reduced the data to one third its original memory consumption at the cost of a 10% performance penalty. Rather than impact customer satisfaction with slower performance, LPL decided to free up hardware budget to increase system memory and see immediate results.

## Technologies & Tools

### Environment

- SQL Server 2000 (TB+ of data)
- .NET 1.1
- .NET 2.0
- .NET 3.5

### Languages

- C++
- COM
- COM+
- C#
- VB.NET
- ASP.NET
- ADO.NET

### Code Tools

- Visual Studio Team System
- Windows Services
- Windows Communication Framework
- .NET Remoting
- Visual Studio 2005
- Wiki
- Visio
- Star team Source Control

- Test Director Defect Tracking
- SharePoint