Software Engineering Radio Podcast: LMAX Architecture

The Software Engineering Radio podcast I chose to analyze is titled, “Mike Barker on the LMAX Architecture” published by the IEEE Computer Society in 2016. Barker is a Research and Development programmer at the London Multi-asset Exchange (LMAX). In this podcast, Barker speaks about the LMAX Exchange system. He highlights the architecture, focusing on how the system works, the important non-functional requirements, and the components of the system. Although most of the podcast was new knowledge to me, I agree with with the authors’ statements about mechanical sympathy and improving system performance. My stance on this topic is that performance of a system and mechanical sympathy are extremely important for developers to keep in mind in order to create exceptional software systems.

Barker begins the podcast by explaining how the LMAX exchange started up. “Initially, there was BetFair, a peer-to-peer sports bidding platform where you could bet on horses and such. However, a faster exchange system with 100 times the speed was needed [1].” He explains that this gave birth to TradeFair, and ideas from this project lead to the creation of the LMAX exchange. Barker explains that the typical business case for LMAX deals with the “goal of building a contract for difference (CFD) in order to trade any asset [1].” He goes on to talk about how the exchange system actually works. “There are four main concepts in the system: instructions, orders, trades, and executions. An instruction is when someone wants to place or cancel an order. Executions occur when instructions arrive at the vendor and are executed. Orders are the entities living within the exchange, and they have a quantity and price. Finally, a trade is when two parties exchange something [1].” He states that the exchange is responsible for ordering orders based on time of arrival, and price placed. “For orders to sell, highest price offered comes first. For orders to buy, lowest price offered comes first. This way, you’re always looking at the best price on the market [1].”

The next big topic Barker discusses is the important non-functional requirements for the LMAX system. He states that response time is the most crucial as it deals with the issue of risk. “If someone places an item for sale, the faster they get confirmation that the item has been placed, the less risk they face [1].” Along with this, the latency is also kept as low as possible. He states that “initially the response time for the system was about 1 millisecond; however, chasing lower and lower latencies lead to a response time of 100 microseconds [1].” He states that “having data all the way over in one place, and logic all the way over in another place is like trying to eat your dinner with a pair of top-sticks through a litre-box [1].” He claims that bringing data up closer to the logic will help increase performance. Also, he states that “knowing about how a cache architecture works, and when to use temporal and spatial locality can help improve throughput [1].” I agree with these claims made by Barker as I believe that optimizing the performance of a system is crucial in any application or domain. I acknowledge that having constantly referenced resources, and recently referenced resources close to the CPU is beneficial and can save time. Even Rick Bunt from the University of Saskatchewan states that locality “has great returned value, and can help create productive applications across a wide range of domains [2].” Another non-functional requirement Barker addresses is the availability of the system, and its ability to handle failover. He states its importance, but doesn’t go into much detail.

The final topics that Barker discusses are the components of the LMAX system and the concept of mechanical sympathy. He states that the system has two main parts: the broker and the exchange. “The exchange is responsible for receiving instructions, managing orders, and matching orders to create trades. The broker is responsible for managing risk. If you buy $1000 worth of carrots, and the prices of carrots go up/down, the broker determines and tracks whether you’re in a profit/loss [1].” As for mechanical sympathy, Barker explains that the term comes from Formula 1 racing. “The idea is not that a driver or developer needs to know everything about building and assembling a car. They just need to have a degree of sympathy or knowledge about how it works [1].” I agree with this idea because I feel that having this knowledge allows engineers to intentionally develop effective software systems. It is not required that everyone knows exactly how something has been developed, but it is important that everyone knows how it works and how to fix problems as they arise.

Overall, I found this podcast interesting and was able to learn a lot of new things from it. I agree with most of the claims made, and feel that they are very applicable. Throughout the talk, Barker mentions non-functional requirements and quality attributes such as performance and availability: crucial topics in software architecture. A minor topic that Barker discusses that I didn’t mention above has to do with testing. He mentions that automated testing helps test the usual code, allowing testers to focus on the interesting topics, and trying to find situations wherein the system doesn’t function properly. In a study done at the University of Warsaw, the authors state that unit testing is an important facet of software quality assurance, simplifying the diagnosis of software flaws [3].” Thus, this material has great relevance in the field of software architecture and design and the points made in this podcast must be noted and utilized.

References

[1] SE-Radio. “SE-Radio Episode 254: Mike Barker on the LMAX Architecture.” Internet: http://www.se-radio.net/2016/04/se-radio-episode-254-mike-barker-on-the-lmax-architecture/, Apr. 12, 2016 [Jul. 10, 2016].

[2] R. Bunt, C. Williamson. “Temporal and Spatial Locality: A Time and a Place for Everything.” Internet: www.cs.usask.ca/faculty/bunt/presentations/Locality.ppt, Dec. 6, 2003 [Jul. 10, 2016].

[3] K. Iwanicki, P. Horban, P. Glazar, K. Strzelecki. “Bringing Modern Unit Testing Techniques to Sensornets.” ACM Transactions on Sensor Networks, vol. 11, Feb. 2015.