FIVE THINGS OPEN-SYSTEMS DEVELOPERS SHOULD KNOW ABOUT THE MAINFRAME

As applications span more platforms, the days of open-systems and mainframe developers sitting in silos on opposite ends of the office are coming to an end. Enterprises need their developers to understand concepts and code tied to both front-end and back-end systems.

So, what should these developers learn about each other? A multitude of stark technical and terminological differences exist between the two computing worlds, for lack of a better term, but in this two-part series we'll just focus on a few high-level concepts.

Go here to read what mainframe developers should know about open-systems. Here, we'll explore five things open-systems developers should know about the mainframe.

1. Mainframe application development is highly specialized.

Mainframe developers tend to be highly specialized, but even so called mainstream developers can learn to manage mainframe applications and data, especially with modern tools and processes that help demystify the mainframe environment. Without these modern tools and processes, learning curves for large mainframe applications can be significantly longer than distributed applications for those who aren't familiar with the mainframe.

The sheer size of mainframe applications are also why many mainframe developers focus on the maintenance or extension of applications—brownfield development—rather than new

development—greenfield development.

Most of this is done in COBOL, a programming language created in 1960. For all the criticism it gets, it's a great language and the best at what it does—run the mission-critical assets of the world's largest organizations, which, in turn, run the global economy.

There are over 220 billion lines of COBOL in the world, with five billion added each year. It's updated frequently, and IBM has even started using Continuous Delivery to send optimizations and feature enhancements derived from customer feedback into the language; however, despite its wide use and growth, COBOL is infrequently taught in computer science programs today, leaving much of the knowledge transfer to experienced mainframe developers in the private and public sectors—on the plus side, those new to COBOL can learn from the best, on the job.

2. Mainframe applications are large and complex.

Mainframe applications are composed of hundreds or more programs that work together but are typically changed and compiled as individual units. Their execution is controlled via JCL (short for Job Control Language), and they can execute either in batch or via an online transaction system like CICS or IMS, while open-systems developers are used to Java Enterprise Edition (JEE) and other application servers. On the mainframe, CICS and IMS both provide services for managing transactions.

IMS is really three sets of capabilities: IMS Transaction Manager; IMS Queues, which hold messages that the Transaction Manager processes; and IMS Databases, which provide fast, hierarchical data access.

Mainframe programs "throw exceptions" that are usually caught by z/OS, the IBM mainframe's primary operating system, and converted into abnormal terminations called abends (Abnormal End). Common batch abends are S0C1, S0C4 and S0C7. Common CICS abends are ASRA, AFCY and others. Abnormal End actually stops program execution, unlike Java Exceptions, which allow the error condition to be caught and handled by the program.

3. Mainframe applications and data are highly secure.

Things are much more locked down on the mainframe than they are in open-systems. Application access is assumed to be restricted and is granted at a very low level through Resource Access Control Facility (RACF). While it limits the openness of the mainframe, this practice also supports the mainframe's reputation as the most secure enterprise system.

Mainframes are incredibly reliable and secure, with built-in hardware and operating-system security capabilities that have evolved over decades to protect applications and data in an evolving technological world. New enhancements, available on the IBM z14 mainframe, enable greater security through pervasive encryption of in-flight and at-rest data.

Due to the esoteric nature of mainframes, they are rarely directly hacked. The easiest way to hack a mainframe is from the inside, and that's becoming a greater concern. Ultimately, you need to secure your mainframe at various levels, including the system and applications, and make it easy to detect and analyze suspicious user behavior, especially as more non-mainframe applications tap into the mainframe to leverage its applications and data.

4. Mainframe application development impacts front-end performance.

On the mainframe, good performance is a necessary goal. Performance, for both online and batch applications, has long been a critical issue, especially since much original development was done decades ago, when hardware was much, much slower than it is today.

Today, the mainframe plays an even bigger role in ensuring applications perform well for end-users. When a mobile banking application fails to complete a request, no one assumes a mainframe program is abending behind the scenes. Yet, checking your account balance, reserving a flight, purchasing things on Amazon and more all require the application you're using to interact with a mainframe.

Good mainframe performance is also critical because poor performance increases cost to your bottom line by increasing your MSUs (million service units—a measurement of the amount of processing work a computer can perform in one hour). The rolling four-hour average of the peak value for the month is how most IBM mainframe customers are billed. If the mainframe back-end application is using more CPU, it will result in more MSU and consequently a higher rolling four-hour average. A higher peak means a larger bill from IBM at the end of the month.

Because the mainframe and systems of engagement are so integrated, and are becoming more so, it's necessary for open-systems developers to understand how their code impacts the performance of front- and back-end systems. You need all systems performing at peak agility, both for customers and economic efficiency, so focusing on improving mainframe performance alongside front-end performance is critical.

5. Mainframe application development is turning toward DevOps.

Mainframe application development generally adheres to an older, more established and stagnant culture of Waterfall that has been ingrained in the industry and its people, processes and tools.

Fortunately, change is in the air. Some mainframe teams are ditching their waterfall-based cultures, demolishing silos and collaborating more with the rest of IT. They're refactoring their COBOL programs, REST-enabling them and using iterative processes and modern tools to develop and deliver innovations more frequently.

From these changes that are shifting the mainframe from a "slow" delivery model toward a "fast" state of operation on par with other systems, the future of an IT organization consisting of DevOps artisans who collaborate across mainframe and open-systems is invariably ahead of us.

The days of mainframe and open-systems developers existing apart from one another are quickly coming to an end. As more web and mobile activity engages the mainframe, IT organizations must transition to more integrated, cross-platform DevOps models that enable mainframe and open-systems teams to collaborate and develop side-by-side.