August 2001

Java in the Element Management System (EMS)

Traditional Approach

The traditional approach in developing EMS software in the 90s was to use C++. The C++ language provides the object-oriented features which allow development of large scale applications with C-like execution speed. The use of Java in the EMS has been mainly restricted to developing the Graphical User Interface (GUI) clients that replaced the difficult to program and non-portable X-Windows. Today, Java is being widely used for server applications throughout the industry. This paper explains why Java is now the language to use for EMS development.

Java Language Advantages

The advantages of Java over other languages such as C++ have been well documented. To summarize, they are:

- **Simple**. The Java language has a syntax similar to C/C++. However it was designed to be simpler than C++ by eliminating rarely used, poorly understood, and confusing features such as operator overloading and multiple inheritance.
- **No Memory Leaks**. A major problem in C++ programs are memory leaks. Java eliminates this problem by providing automatic garbage collection. Whenever an object is no longer used (or referenced), the object and its memory will be reclaimed automatically.
- **Robust**. Java does not have the same concept of pointers that C++ has. As a result, Java eliminates the problem of bad pointers which overwrite memory and corrupt data.
- **Portable**. Java code is compiled into bytecodes which are run by the Java Virtual Machine (JVM). Thus, Java programs run without change wherever the JVM is available. Contrast this with C++ where the code is not 100% portable and must be recompiled on each different hardware platform and operating system.

- **Multithreaded**. Java has built-in support for threads and synchronization primitives. The Java threads are typically mapped to native OS threads by the JVM. Otherwise, the JVM manages the Java threads itself. C++ has no support for threads and must use OS-dependent thread libraries.
- **Consistent Feature Set**. Java is an industry standard from Sun. The features are standard and consistent across platforms. Compare this to the state of the C++ language which has been standardized finally, but the compiler implementations lag the standard. The C++ standard added many useful new features that are not supported by the compiler or have high overhead or performance issues (e.g. Exceptions and RTTI). As a result, these features must be avoided due to these problems and the non-portability of the code due to differences in compiler support.

Advantages of Java in the EMS

Now that we have discussed the general advantages of Java, we will explain how Java provides advantages in implementation of the EMS.

✦ Faster Time-To-Market

Vendors and telecomm startup companies are under tremendous pressure to develop their equipment and management system in a very short amount of time with limited money to do so. Using Java to build the EMS provides a *significant* reduction in EMS development time. The reasons for this speed of development:

1. **Easy to Use**. The Java language is easy to develop with. The syntax is similar to C and C++ making it easy for developers new to the language to be productive.
2. **Simpler Code**. Code written in Java is simpler and less complicated than comparable code in C++. The developer does not have to worry about freeing memory and is provided with a large standard library to use. Simpler code also translates into easier maintenance.
3. **Rich Set of APIs**. The Java 2 Standard Edition (J2SE) platform provides a rich set of libraries and standard classes – 1,520 classes in 59 packages covering networking, file I/O, collections (e.g. linked lists, hash tables, sets), and GUI components. Developers do not have to spend time "re-inventing the wheel".

✦ Reduced Cost

The cost for development and maintenance of the EMS is reduced because of the speed of development described above. In addition, because Java is portable, you do not incur costs for porting your software to different platforms. Portability is not guaranteed; all software including the JVMs have bugs. This means that

testing of your software is still recommended and required for each platform you support.

✦ Free

Java is free for commercial use. The development tools and runtime environment are freely available and supported on all of the major platforms including Sun Solaris, HP-UX, MS-Windows, IBM AIX, and Linux.

✦ Portability

Java is a portable language and is supported on all of the typical EMS platforms including HPUX, Solaris, and Windows. This means that your EMS software can run without modification on different vendors hardware. Now you are not locked in to your hardware vendor.

The other major lockin occurs with the Database vendor. Java helps with this by providing a generic Database API called JDBC, which allows your software to be independent of the actual DBMS.

Finally, Java is an excellent choice for developing a portable EMS GUI client. The Java Foundation Classes (JFC), commonly known as Swing, provide a portable GUI toolkit that do not rely on the native OS windowing system. Thus, the GUIs developed have the same look and feel on different platforms (e.g. Windows and Solaris).

✦ Built-in Distributed Objects

Java provides built-in support for creating architecture independent distributed applications. Java provides two object-oriented RPC-like mechanisms which can be used to build remote client/server communications: RMI and CORBA. Remote Method Invocation (RMI) is an easy to use mechanism for building a server object. The remote interface is defined in a standard Java interface. CORBA is a standard from the OMG which supports mappings for many programming languages including Java. The remote interfaces are defined either in Interface Definition Language (IDL) or using a Java interface which is converted to IDL. Using RMI and CORBA, the EMS software can be distributed into multiple servers running on the same machine or different machines.

✦ Component Architecture

Java provides a component architecture called Java Beans. This allows developers to create reusable software components which can be plugged into an application and customized through the bean's API. The advantage of this in EMS development comes through (1) buying third party Java Beans and using

them in the EMS software, and (2) developing your own internal EMS beans which can be reused for other applications.

✦ Enterprise Architecture (J2EE)

Sun provides an Enterprise version called Java 2 Enterprise Edition (J2EE) which provides an additional standard set of APIs including:

- Enterprise JavaBeans
- Servlets and Java Server Pages (JSP)
- Java Messaging Service (JMS)
- Transactions
- XML

J2EE allows the EMS to be built using Enterprise JavaBeans which run in an Application Server environment. The Application Server provides support for scalability, load balancing, and transactions.

Why Not Java

With all of these advantages that have been described above, what would be a reason not to use Java? The main concern with Java has been its execution speed. The language is compiled into byte codes which are interpreted by the Java Virtual Machine. Java can probably never be as fast as comparable C++ code. However with advances in JVM technology and techniques such as the HotSpot VM and Just-in-Time (JIT) compilation, Java can achieve comparable speeds.

Conclusion

Java is the language of choice for developing an EMS, because of faster-time-to-market, reduced cost, portability, and state-of-the-art features. QCOM itself has successfully built many EMS systems completely in Java; all meeting the customer's performance and capacity requirements. For these reasons, QCOM provides the FasTrack ComponentSuite all written in 100% Java.