JCP Position Statement for ME

James Hunt, CEO aicas GmbH

Java Micro Edition has been lagging behind the development of J2SE for some time now. This has a negative impact on the compatibility between J2ME and J2SE inhibiting the sharing of code between the desktop and embedded worlds. Though we at aicas have worked hard at bringing the write-once-run-everywhere concept to realtime and embedded systems through the support of J2SE libraries for these system, this does not replace the need for smaller subsets of Java for application that require dynamic extension through downloadable apps running in restricted environments. Therefore, I would like to support both incorporating J2SE 1.6 language features in Java ME and improving the compatibility between J2ME and J2SE.

The Java Community Process needs to promptly update the core ME APIs for both CDC and CLDC to be 1.6 compatible. CDC should eliminate class level subsetting for classes that are taken from J2SE. JSRs which provide new APIs for J2ME that are not in J2SE should provide compatibility specifications for J2SE as well. Ideally, the reference implementation should run both in j2ME and J2SE. I would like to help facilitate this development through constructive interaction with the JSR specification leads.

Despite the work towards compatibility, the JCP should not loose sight of the fact that the requirements for embedded systems, particularly realtime systems, are often stricter than that needed for the desktop. The best work in this direction is the Real-Time Specification for Java (RTSJ). Though the full RTSJ specification is beyond the scope of most desktop implementations, there are some key features, such as asynchronous event handlers and stricter definitions of synchronization behavior, that would aid the general Java programmer. Some of these features should be migrated into the core Java classes and care should be taken not to reinvent features that already exist in the RTSJ.

The long term viability of Java as a commercial development language depends on maintaining the highest possible level of compatibility without loosing the flexibility needed to address the multitude of application domains that could benefit from Java technology.