



Eclipsing all Others

Socialized Software IDE Makes it Big

There was a time when the device software tools business was simple. Life was good. Proprietary environments from different vendors competed for the hearts and minds of software developers by boasting about their robust feature sets, their remarkable performance, their broad platform support, and their competitive price tags. Titanic niche warriors like Wind River, Green Hills, and Accelerated Technology battled in the brains of embedded developers for build and debug supremacy. The embedded development world was an island unto itself, and the industry liked it that way.

Then trouble came. Over in the desktop and enterprise development worlds, it seemed, a revolution was afoot. The dominance of the desktop development environment of Microsoft had sewn the seeds of a socialist revolution. A not-so-secret society had spawned a groundswell of collaborative development among several companies and independent software developers who wanted to create an alternative for themselves – to take control of their own development destinies. Following in the footsteps of the Linux phenomenon, Eclipse was born.

Conceived in late 2001, Eclipse arrived as the alternative to the ubiquitous Microsoft Visual Studio series. It is an open-source integrated development environment (IDE) based on open-source development principles similar to Linux. Eclipse is described as “an open source community whose projects are focused on providing a vendor-neutral open development platform and application frameworks for building software.” Championed by an organization called Eclipse Foundation, Inc., the original incarnation of Eclipse came from IBM, who released their Eclipse platform into open source and worked with other companies including Borland, MERANT, QNX, Rational, Red Hat, SuSE, TogetherSoft and Webgain to form a consortium overseeing the further development of the platform. In 2004 the Eclipse Foundation was organized into the not-for-profit corporation that continues today.

"Growth of Eclipse in the device software tools area has been very rapid," explains Mike Milinkovich, executive director of the Eclipse Foundation. "We have seen tremendous success with tool vendors joining Eclipse and with users adopting the tools." Eclipse projects serving the embedded and device software space include application builders for GUI development, enhanced debug models for device debugging, and target management frameworks to facilitate management of remote systems.

Talking to developers that contribute to the Eclipse effort, one frequently hears explanations of the form: "I never wanted to develop another _____ again," where the blank is filled in by that developer's too-many-times-in-a-career project, like a runtime debugger for example. Chatting with companies that donate engineering resources to the project, the explanation typically takes the form: "We wanted to be able to focus our development resources on our unique value-added capabilities such as _____," where their blank was filled in with something other than the typical capabilities of an IDE.

The first big Eclipse waves to hit the embedded space landed with a remarkably happy reception at embedded development tool suppliers that had previously made a living on highly proprietary, tightly integrated development suites. Embracing Eclipse required a fundamental philosophy shift in those companies and necessitated a complete overhaul of their marketing messages and product development processes. Suddenly, loads of relative minutia that had been previously cited as differentiators for one IDE or another were off the table. Tool providers now had to focus their competitive efforts on more substantive issues.

With Eclipse, everyone started with an essentially level playing field. Companies that specialized in tools specifically for embedded developers were compelled, if not almost required, to focus their development resources on creating value specific to those embedded developers. "Cost of doing business" development in commonplace, pedestrian areas of IDEs came to a halt.

Doug Gaff, manager of device software development platforms at Wind River, describes the infiltration of Eclipse into the tool developer community. "First, the engineers don't want to look at the open source framework. They say, 'We've got our own stuff, and it's better.' Then, after awhile, they look again and admit, 'Hey, there are actually some good things on here and we can build on top of them.' After awhile, they're hooked. Then, as they build on top of Eclipse, they start to identify holes – places where important capabilities for embedded

design are missing. The best way to fix that, of course, is to contribute your work back to the project. Companies gradually transition from users of Eclipse to developers of Eclipse.”

Wind River has jumped on the Eclipse bandwagon in a big way, signing up to be a “strategic developer” for device-level software platforms. “We wanted to create a place for people doing device-level software to come and work together,” Gaff continues. “CDT [Eclipse C development tools project] is a great place for C developers, JDT [Eclipse Java development tools project] is a great place for Java... We wanted to create an extension with a device software focus.” The project for device software is called “DSDP” for “Device Software Development Platform” (Evidently “DSDT” was already taken by Eclipse’s “Differentiated System Description Table” – no operator overloading here.)

Other embedded and device software development companies have made major commitments to Eclipse, and to the Eclipse philosophy as well. Mentor Graphics (formerly Accelerated Technology), Lynxworks, QNX (as a founder of Eclipse), and many others have integrated and interfaced Eclipse, seeing a combination of a market opportunity, user demand, and decreased development burden.

These companies all face a similar conundrum in their own development efforts. When an important capability is identified that isn’t provided by Eclipse, do you develop it in a proprietary fashion and try to add it to your own products as a value-added capability, or do you develop and contribute the capability to Eclipse? The decision isn’t as complex as it might seem on the surface. “That decision depends upon the size of the change and how central the capability is to our core business,” Gaff explains. “ Wind River builds broad platforms that include lots of capabilities. If there is something missing in the IDE where it doesn’t do what we’d like, and that capability is not in our core value-added area, we’d contribute it. On the other hand, if we needed to add a brand-new capability that is directly beneficial to our customers and doesn’t logically fall within the Eclipse framework, we’d look at developing that as a product. It’s usually pretty obvious which way we’d want to go.”

Eclipse has moved so rapidly into the embedded space that it is now competing with Visual Studio as the most used software development environment. Over time, it seems that the scope and penetration of Eclipse in the embedded community will only increase. That means more capable, more robust development tools for us, allowing us to focus our efforts on creating new

and innovative embedded products instead of wasting our time evaluating alternative development tools.

Kevin Morris, Embedded Technology Journal

May 9, 2006