

# EDN<sup>®</sup>

JAN **24**  
 Issue 53/2008  
[www.edn.com](http://www.edn.com)

VOICE OF THE ENGINEER

pulse



## VOICES Green Hills Software's David Kleidermacher

**D**avid Kleidermacher is the chief technology officer at Green Hills Software, a company that provides high-performance compilers, software-development tools, and real-time operating systems for developers of embedded systems. Recently, during the Green Hills embedded-software summit, he answered a few questions about the two main topics of discussion at the summit: security and virtualization. The following is an excerpt of the interview. For the complete version, go to [www.edn.com/080124p1](http://www.edn.com/080124p1).

**What are the top challenges your company is focusing on to try to help your customers with their design efforts?**

**A** Security is a top concern: How can we provide tools that help our customers address security—not merely from an operating-system perspective, but also from a position of addressing the whole picture? It's not just operating systems but all the things we need to do. We need to do a lot more on a lot of different planes to help people make the software more secure. We have shown that we know how to make our own software secure, but we have not yet fully enabled our customers to make their own software fully secure. From my perspective, it's a multitiered approach; operating systems and virtualization are definitely important, but you need the right runtime environment, the right software-development tools, and the right

process—something we have not talked about much to date. You can talk about formal methods, but there is more to it than that, so that you can address how you go about developing your software so it is more secure and still meets the same time-to-market demands that you have.

Most applications cannot afford \$1000 per line of code to ensure the same level of reliability as the space shuttle; the reality is that most of the software in the world does not need to be at that level. If you have a computer system running Windows, Linux, and application code, it might have 100 to 200 million lines of code running on it. How much of that code is high assurance? Only about 20,000 to 30,000 lines of code of that—say, the kernel and a couple of other special components—need to be fully secure.

The things we've learned about making our own software secure are streaming out in bits and pieces into our product base, but there is room for a lot more, such as in the automated-testing area, coding-standards enforcement, and tools that help you more reliably develop software. There are also the process things, and this area is one we have not historically been pushing much at all, but we are starting to touch more on it now.

**“Virtualization” is a term that the industry is using across multiple contexts, such as design-time and runtime virtualization. Should the industry be using a single term to describe an abstraction concept to apply to multiple contexts?**

**A** The term “virtualization,” when you use it as an abstraction of anything, is too general. I have been pushing to use different terms to distinguish between the contexts because using a single term for all of them is confusing. Design-time virtualizations often refer to hardware-, software-, and co-prototyping tools that act as development platforms before the hardware is ready to support parallel development and shorter design cycles. I have proposed using the term “virtual prototype” to refer to these types of abstractions, and it may

even be appropriate to have different terms for each of these types of prototypes. For runtime-virtualization tools, I like to use the term “virtual machine,” or “virtual-machine monitor,” or even “hypervisor” to apply to runtime virtualization, especially because no one has previously used these terms, especially “hypervisor,” to refer to virtual-prototype contexts.

There is another classification that I use that takes it another step further—beyond design or runtime—to classify the general application areas it is for, such as virtual IT, which is concerned with consolidation and aggregation for better server management or even the flexibility to run different operating systems or hardware platforms. Another category is what I call virtual hybrids, which use virtualization as a tool to enable more flexible computing models. This [approach] is basically a runtime virtualization, but it is not just for the sake of virtualization, as virtual IT is pure virtualization. Virtual hybrids can [involve] running some legacy code alongside some real-time application or some security-critical application. A whole gamut of applications exists, and I think these types of applications deserve their own category because they are so different from your typical VMware or parallel-processing applications.

**Compilers provide an important level of abstraction that enables them to optimize instruction sequencing; is anything preventing compiler technology from addressing systems as they continue to expand the amount of integrated and specialized resources and the number of connected processors in a single system?**

**A** The compiler is starting to do some things to accommodate this [scenario], but the current coding languages hamstringing compilers today, partly because they do not automatically capture concurrency or the impacts of latency and bottlenecks in memory or special processing resources.

—by Robert Cravotta