

eWEEK ENTERPRISE NEWS & REVIEWS

Fusion ZONE
www.fusion-zone.com

From the Market Experts at Ziff Davis
Your ultimate middleware resource

MIDDLEWARE

SERVICE-ORIENTED ARCHITECTURE

IDENTITY MANAGEME

ENTERPRISE PORTAL

BUSINESS INSIGHT

CONTENT & COLLABORA

The Future of Programming: Less Is More

August 28, 2006

By Darryl K. Taft

In the future of the process of building software, the ability to do more with less could become the norm, some developers say.

"The future of programming is less," said David Heinemeier Hansson, the creator of the Ruby on Rails Web application framework and a developer at 37signals in Chicago. "Less configuration, less committees, less proprietary infrastructure. The rise of open source is causing this. Along with this transformation, we'll see the open-source, dynamic languages start displacing their enterprisey counterparts."

ADVERTISEMENT

 **symantec.**

Symantec Data Center Foundation.

Take a tour of the data center and see for yourself >

Hansson's Ruby on Rails is considered a lightweight alternative to the Java Enterprise Edition.

RELATED LINKS

[Race Is On to Woo Next-Gen Developer](#)
[Software Product Lines Automate Development](#)
[Mainframers Learn New Tricks](#)
[Microsoft Playing Games with Developers](#)

"It's no accident that Java is owned by Sun [Microsystems] and C# by Microsoft, while Perl, Python, PHP and Ruby steam from the open-source world," he said. "It's also no accident that the 'WS-deathstar' construction is being fielded by IBM and

Microsoft while the REST [Representational State Transfer] movement is one of grassroots. The age of the vendor is nearing its end. A new age of peers, collaboration and commons is upon us."

"We always have to push to simplify the 'simplifiable'—especially in the face of ever-increasing complexity," said David Intersimone, vice president of developer relations at Borland Software, in Cupertino, Calif. Intersimone will be going with the Borland IDE (integrated development environment) tools division, known as "DevCo," which Borland is about to sell off.

The race is on to woo the next-generation developer. [Click here](#) to read more.

"The DevCo road map takes us out the next three years," he said. "At the same time, being developer-focused doesn't mean that we are limited to Delphi, C/C++/C# and Java programming languages. There are many languages/paradigms/stacks that we have to deal with. Some languages are getting features from others, such as the declarative additions in C# 3.0 data, for example. And Ruby could be

thought of as 'Lisp for the Internet.'"

Intersimone shared his vision of what programming in 2010 might be like.

In 2010, XML will be the fundamental data type for all programming languages and databases, he said. Also, languages will include syntax extensions for proving correctness, unit and system testing, parallel processing, aspects, and declarative and functional programming, he said.

And for tools, in 2010 the development environment will be a container or dashboard connected to the Internet where RSS feeds, design, development, testing and deployment take place, Intersimone said. Moreover, artifact management will be automatic, as will be refactoring, testing, audits, metrics, security verification and reporting. And all IDEs will contain at least 10 different general-purpose and DSLs (domain specific languages) that can be intermixed inside the same project parts, he added.

In addition, Intersimone's vision of development in 2010 has large libraries of reusable distributed objects, and Web services being available for every developer. And all developers will be connected to each other using a global directory, he said.

Sergey Dmitriev, founder and chief executive of JetBrains, in Prague, Czech Republic, said any modern software platform and language should have 12 main features: object orientation; support for language extensions and meta-programming; an intelligent IDE; clear syntax and readable program code; support for collections; support for writing GUI applications; support for writing Web applications; support for multithreading; support for functional types, closures and continuations; garbage collection; an effective virtual machine/runtime; and the ability to deploy to any operating system.

Dmitriev said the platform should allow developers not only to extend an existing language, but also to easily design a whole new language and build an intelligent editor for it.

"To run programs written in such a DSL, the platform should support writing generators to any existing runtime platform—Java or .Net or whatever," he said. "Using such specialized DSLs allows writing programs on a much higher level, so these programs will be much more maintainable and expressive."

Dmitriev refers to this approach to development as LOP (language-oriented programming), and said JetBrains' MPS (Meta Programming System) and Intentional Software's Intentional Software system are two technologies that implement LOP.

"The dynamic versus static languages arguments have a very long history, and I wouldn't say that it is major issue for the future," he said. "I'd say that it is an issue of the present and the past already."

Dmitriev said he expects there to be more expressive languages for knowledge representation.

Next Page: Time to make programming creative again.

However, "in my opinion the main problem that should be solved is making programming a creative job again," he said. "We are doing too much boring coding work now and too few creative tasks."

He said part of the problem is weak tools. "The existing programming languages have not gone too far from machine code; they just use better words for the same instructions," Dmitriev said. But they do not allow for much more than just manipulating with stack, memory and control flow, he said.

"I believe that programs should be written not in the language of computer, but in the language of the problem's domain, and then we'll reach the expected level of simplicity, clarity and maintainability," Dmitriev said.

According to John Crupi, chief technology officer at JackBe, in Chevy Chase, Md., developers of the future will need to adopt a new set of programming skills, including understanding how to create business-grained SOA (service-oriented architecture) services.

"Coming from J2EE [Java 2 Platform Enterprise Edition] land, we're familiar with finer-grained application services," Crupi said. However, "SOA services need to be aligned with business functionality and requires IT working closely with the business unit to correctly define the business services."

Read [here](#) about Microsoft's support for dynamic languages on .Net.

Future developers also must learn Web event driven programming, Crupi said. Crupi is a former Sun Microsystems distinguished engineer.

"Web applications will move more and more toward Web-based events," he said. "AJAX [Asynchronous JavaScript and XML] gives us the ability to initiate asynchronous requests, but not to receive asynchronous push events from the server. This will all change. In the future, Web-based applications will subscribe to business events and be mostly based on this interaction model. This new Web event model requires programmers to program more at the business event level and less at the user event level."

With an academic perspective, Shriram Krishnamurthi, a computer science professor at Brown University, said developers of the future will have to deal more than ever with issues of sharing and access control, handling user concurrency, coping with global scale, and writing statements about programs rather than only writing program statements.

Krishnamurthi said developers rush to use a "lint," or static, analysis tool for every new language they encounter, "but run screaming in the other direction when confronted with abstruse-sounding concepts like verification, abstract interpretation and type theory—even though the latter are really principled means of obtaining the former."

However, part of the blame for this lies with researchers in these areas, who have not done a particularly good job of communicating the connection via usable tools, Krishnamurthi said. But that has been changing in the past few years, he added.

"What these tools really need is more information about the hidden constraints in programs: the structure of the heap, the ephemerality of data, the protocol of locking, the pattern of communication, and so on," Krishnamurthi said. "Inferring these properties is expensive and very brittle, but writing lightweight specifications about them gives tools a great deal to chew on—ultimately making programs both faster and more robust."

Check out eWEEK.com's Application Development Center for the latest news, reviews and analysis in programming environments and developer tools.

Copyright (c) 2006 Ziff Davis Media Inc. All Rights Reserved.