Microsoft: Making Programming Easier

(2008-01-28) - Contributed by Darryl K. Taft

Officials within Microsoft's software unit outline key trends that the company is trying to address, including integrating markup and imperative languages.

REDMOND, Wash. -- A primary target for Microsoft's developer division is the elusive goal of simplifying programming.

Jason Zander, general manager of the Microsoft Visual Studio tool set, spoke at the Lang.NET Symposium Jan. 28, noting that making programming easier is number one among the key trends he sees in the industry, and that Microsoft is working on it.

Zander said he and other speakers at the show would address many of the hot issues developers face today. A key issue is deciding whether to "go for reach or a rich environment," he said.

Making Web programming easier is another goal, as is the integration of query logic, which Microsoft is providing with its LINQ (Language Integrated Query) technology. Yet another trend Zander said he sees is easing the integration of markup and imperative languages.

Developers also are concerned about support for parallel computing advances.

"There's been a lot of research over the last 20 some years and the question for us is how we're going to make that [programming for parallel environments] easier," Zander said. "Computer science is one of those fields where you can see what's going to be cool by looking back 20 to 30 years."

Meanwhile, Anders Hejlsberg, a Microsoft fellow and lead on the C# development effort, said the C# project started nine years ago this month and is now at version 3.0. Version 1.0 of C# was all about "getting the world to managed code," he said.

C# 3.0 "was our first real chance to step back and look at language evolution," Hejlsberg said. He said the biggest feature in C# 3.0 is LINQ.

C# 3.0 features include local variable type inference, lambda expression and expression type, extension methods query expression, automatic properties and anonymous types, Hejlsberg said.

Zander said the goals of the CLR (Common Language Runtime) included modernizing Microsoft's programming interfaces, creating a consistent programming interface across the board to "stop building tons of duplicated runtime functionality" and forms packages, to allow developers to use all their skills orthogonal to language choice, and to improve productivity.

He also showed a memo from 1998 displaying a work item list that included adding features such as integer and float support, a just-in-time compiler, multi-threading, loader support for multiple DLL and support for calling native code.

{mospagebreak title=Breaking Down Walls}

He said that version was successful because it delivered a full type system, full class libraries and full tools support. Version 2 of the CLR brought with it the edit and continue feature and full generics support.

Overall, the improvements have been built on previous investments. For instance, LINQ and the Silverlight cross-platform plug-in for delivering next-generation media experiences and rich interactive applications are examples of technologies

that have benefited from previous investments.

"Procedural, functional and dynamic languages were a goal from day one," Zander said. And version 2 of the platform also added greater performance and lightweight code generation.

Meanwhile, the DLR (Dynamic Language Runtime), which is based on the CLR, allows for easier integration of other languages in the future.

Zander also discussed how the Microsoft developer division evolved some of its licensing to include open-source-style licensing, beginning with the SSCLI (Shared Source Common Language Infrastructure) to the Microsoft Reciprocal License to the Microsoft Permissive License, which has "made it easier for future releases" to go out under open-source like licenses.

According to Zander, Microsoft's ASP.NET AJAX, IronRuby and the DLR have been released under these licensing terms.

Meanwhile, Hejlsberg said the walls between various programming styles are breaking down.

"The taxonomies of programming languages are starting to break down," he said. Hejlsberg said there are dynamic languages, programming languages and functional languages. In addition, "future languages are going to be an amalgam of all of the above. I think that bleed will continue. We're very keen to exploit that in C#."

However, Hejlsberg said there are things missing in C#, "like dynamic linking, and we're planning to add that."

Hejlsberg also said his group and others at Microsoft are working on solutions to deal with concurrency or programming for parallel environments.

"PLINQ [a parallel implementation of LINQ] is an example, but we need to look at more ways," Hejlsberg said. "We have to find meaningful ways to write concurrent programs or we're out of business."