
Computer Science Colloquium

Where Has My Compiler Gone?

Dr. Jim Miller
Microsoft

Thursday, 30 March 2006

Olsen 311

Refreshments at 8:30, Talk from 9:00-10:00

Over the past 50 years the compiler's role in a software system has changed dramatically. From an engineering point of view, this means that the requirements on a compiler have changed and it's sometimes hard to recognize a compiler anymore. We'll discuss the relationship between a compiler, a runtime, a development environment, and an operating system. We'll talk about how the traditional front-end/back-end split in a compiler can be exploited, and how this allows us to create a number of variations on the traditional compiler that are tailored for different purposes. We'll talk about how different views on the purpose of a high-level programming language result in different language designs and compilation strategies. And we'll see that there are compilers hidden in some unusual places these days.

Bio: Jim Miller is a senior architect on Microsoft's Common Language Runtime (CLR) team. His current work is on architectural changes to allow innovation in the core of the CLR and the managed Frameworks while preserving backward compatibility. He also serves as liaison with the academic, research, and compiler communities for the team.

Jim holds a PhD in Computer Science from MIT and served on the faculty at Brandeis University as well as on the research staff at MIT. He has been on the research staff at Digital Equipment Corporation and the Open Software Foundation. Before joining Microsoft, he was on the senior management team of the World Wide Web Consortium, reporting to Tim Berners-Lee and in charge of work on security, electronic commerce, child protection, privacy protection, accessibility, and intellectual property protection.

Jim joined Microsoft in 1998, leading the program management team for the kernel of the .NET Common Language Runtime (CLR). His responsibility included garbage collection, metadata definition and file formats, intermediate language (IL) definition, IL-to-native code compilation, and remote objects. He also serves as editor for ECMA TC39/TG3, which is charged with creating an international standard for a Common Language Infrastructure. To validate this standard, Jim helped create the Shared Source CLI (also known as Rotor), a complete implementation of the standard, runnable on Windows, Macintosh, and Unix operating systems, available in source form for teaching and non-commercial purposes.