Foreword to the Second Edition

Wow. It has been a long time since I wrote the foreword for the first edition of *ATL Internals*. Reading through the old introduction really takes me down memory lane; I can hardly believe that it has been almost eight years. Not long after I wrote it, I moved on to the Windows team at Microsoft and then on out of Microsoft a year later. I came back to Microsoft (and the Visual C++ team) a few years ago, and I am now managing several development teams in Visual C++. One of these is the libraries team, of which ATL is a part, and it is fun to be involved in ATL again. Jan and Christian have both moved on, although Nenad expanded the windowing classes from ATL that I mentioned in the first introduction into a separate library called WTL (Windows Template Library[1]). WTL is now a Microsoft open-source project that Nenad manages.

ATL has changed in ways I never could have predicted, and it has been bittersweet to see it continue to grow without being personally involved. There have been many great people who have worked on ATL over the years. Some of them I have known quite well and others I never knew.

When I mentioned "some new ways of accessing the ATL functionality" in the first foreword, I was referring to attributes. This technology was delivered in Visual Studio .NET 2002, but it never really developed into what we envisioned. ATL attributes still work in the current release and they can be quite powerful, but there are no plans to expand their use. This new version of ATL Internals provides lots of updates and does cover attributes, but doesn't assume that you're going to depend on this feature. This edition also includes a very nice introduction to ATL Server, which provides a flexible, high-performance way to create web applications. If performance is a critical requirement, ATL Server was built for you. Other ATL 8 improvements include better security, full 64-bit support, better scalability, debugging improvements, support for C++/CLI, and managed ATL components.

What has become the .NET ecosystem was just getting underway back in 1998. It has revolutionized programming for many developers and will continue to deliver improvements in the years to come. However, COM programming (and

[1] http://wtl.sourceforge.net

ATL) is still very much alive and is very important to many developers both inside and outside of Microsoft. The second edition of this book, like the first, provides the details you need to maximize your investment in those technologies.

Jim Springfield
April, 2006

Foreword to the First Edition

When I first saw the title of this book, I told Chris Sells that it sounded like the book I always wanted to write. Ever since we released ATL, some of us have been saying, "We should write a book on how ATL works." After reading *ATL Internals*, I don't think there would be much left for me to write about. Actually, this is kind of a relief. At this point, I think most aspects of ATL have been covered, and *ATL Internals* provides an excellent source of information of the inner workings of ATL. So, Chris asked me to provide some information that can't be deduced by looking at the ATL source code.

A Brief History of ATL

I first got into templates in late 1995, while I was a developer on the MFC team. A friend of mine here was evaluating various STL vendors for the Visual C++ product, and he talked to me a lot about templates. I played around with templates a bit, but didn't do much with them. Soon after, the VC team split off an enterprise team to focus solely on Visual C++ 4.2 Enterprise (our first VC enterprise product). I moved over to head up the libraries work for VCEE. At the time, we explored several different ideas. Microsoft Transaction Server was just getting started then, and we talked a lot with them about COM, transactions, databases, and middle-tier business objects. Pretty quickly, we realized that we needed a better mechanism for creating COM objects from C++. Jan Falkin and Christian Beaumont were working for me at that time. Jan was working on an automation interface to ODBC data sources, and Christian was working on a template-based access method to ODBC data sources (a forerunner of our current OLEDB consumer templates). I was working on the COM infrastructure, since everything was already pointing to COM back then.

Initially, I was just playing around with COM and templates, but slowly it started to stabilize around a few important concepts. From the beginning I wanted to support all threading models, but I didn't want to pay for it unless it was needed. The same was true for aggregation. I didn't want anyone to have an excuse not to use ATL. (I didn't want to hear, "Well, ATL is cool, but I can save a couple of bytes if I do this myself.") So, performance and flexibility came before ease of use when

that decision had to be made. One of the main concepts that came out of this was that the class a user writes is not the class that was actually instantiated. This allowed many optimizations that otherwise could not have occurred. Some of the other concepts were multiple inheritance for interfaces, "creator" functions, and a data-driven COM map. We started to show this around and got a lot of good feedback on it. Several people thought we should get this out to customers as soon as possible, so we decided to RTW (release to the web) in the early summer of '96. That was ATL 1.0. Our working name for our libraries was MEC (Microsoft Enterprise Classes), but our marketing person thought we should have something that more reflected what we were doing. Because of our COM focus and the fact that at the time, everything was being called "Active" something or other, we selected the name *Active Template Library*. We got a good reception for it and in late summer of '96 we released ATL 1.1. By this time, Jan and Christian had started working directly on ATL. ATL 1.1 had bug fixes and support for a few more features such as connection points, NT services, RGS registry support, and security.

After ATL 1.1, we started working on ATL 2.0. Its primary focus was the creation of ActiveX controls. Jan and Christian did much of the work on this, while I still focused on the core stuff (such as rewriting the connection points to make them smaller). Nenad Stefanovic also joined us at that time and started work on the windowing support in ATL, as well as doing the composite control support in VC 6.0. We were originally planning on ATL 2.0 to be shipped on the web targeting VC 4.2. However, our plans changed and we changed ATL 2.0 to ship in VC 5.0 (12/96), and shipped ATL 2.1 with the Alpha version of Visual C++ 5.0. The only difference between ATL 2.0 and 2.1 were some bug fixes for Alpha, MIPS, and PowerPC. We also simultaneously shipped ATL 2.1 on the web with AppWizard and ObjectWizard support for VC 4.2. After a couple of months of working on ATL 3.0 (called ATL 2.5 at the time), Christian and I were burned out and took some time off from ATL, while Jan took over as ATL lead. A few months later, we came back, and Christian became the ATL lead while I moved on to explore some other things for Visual C++, although I do still get into the source code every now and then.

We shipped VC 6.0 in June '98 and are currently working on the next release. Expect to see lots of cool new stuff in ATL as well as some new ways of accessing the ATL functionality. I am glad to see ATL continue to evolve, while at the same time maintaining the original goals of generating small, efficient code. So, take a look at this book, learn some new tricks, and gain a deeper understanding of how it all works.

Jim Springfield
October, 1998