Having a hard time to choose between C# and JAVA … IKVM can bridge between these languages

June 13, 2009
Some time ago I was working on a project where a new Java based platform was being integrated with an existing message bus, already connecting up C# based services, to create a B2B communication chain between several companies. The Java platform was appliance with a Java application server on it.

In the original architecture we had planned to run a message translator on the appliance. Because of several technical integration issues we got way behind. We ended up in a situation where a team already spent a considerable amount of time and budget on developing mappings for the translator. The same translator we had trouble with integrating into the B2B communication chain.

So, on the one hand, finding another translator was no option because of the work already done on the mappings for the translator. On the other hand, we hadn't another Java appliance to host our translator.

Looking for a solution I investigated the possibility to host the Java translator in a C# based service and move it to the existing part of the communication chain until we had the integration problems solved.

At first I was very disappointed in finding no easy way to do cross-language interoperation. The best I could come up with was spinning off a process from C# code that would run the Java translator in a Java sandbox for every message that would pass through the message bus. No need to say this was a performance killer!

I kept looking and finally bumped into IKVM (http://www.ikvm.net). **It allowed me to turn the translator java classes into a .NET C# library**. Once I got the library, creating the C# code to host the translator was still quiet complex but the performance gain was enormous.

IKVM's magic is situated under the bonnet were Java byte code is turned into C# intermediate language and vice versa. I used to *ikvmc* compiler to turn the translator's JAR into a .Net dll by means of *target:library* option. This compilation is the power of IKVM because at run-time no time is lost to do complex inter-language interpretation: *compile once and (re-) user everywhere!?*

I must say I wasn? expecting any result because these kinds of conversions aren't trivial. Most of the time I?e seen it work for basic Hello World examples but fail for anything with a realistic complexity. I was really amazed finding out that IKVM was up to the job!

It is worth noting that IKVM is still under development and the newest release is 0.4 so still an alpha version. Personally I don't care because it really saved my day.
