Why OpenArm?

1. Why bother?

This site's index page provides a pretty good explanation of what OpenArm is (or, at least, what it intends to become). But it doesn't answer the question "Why?" Why bother implementing the Open Group's ARM specification in Java?

The short answer, of course, is "Why not?" Flippant one-liners don't get us any closer to an explanation, however. The explanation itself is really quite simple: because there is no other freely available implementation of ARM.

ARM itself is a pure specification. The Open Group does not provide a reference implementation -- they don't provide any kind of implementation at all. Traditionally, they've left that for the various commercial vendors who are members of the working group. And there are a number of commercial, proprietary implementations of ARM, notably from Tivoli and HP. These implementations are fairly expensive, however.

The market hasn't complained about that, however. The typical "user" of ARM -- an organization which uses it to instrument an application to gather performance information -- is a Fortune X000 company. These folks are accustomed to shelling out large sums for "enterprise" class software, and the vendors have seen no reason to package or market their ARM tools as anything but.

I have reason to complain, however. I had the opportunity to work with ARM at a major bank, and was involved in both evaluating tools and instrumenting applications. Although I've come to regard the ARM API as a bit "hidebound" by its procedural, C-oriented roots, it's nonetheless THE standard in this problem domain -- the only one of its kind that I'm aware of. Yet its use is limited to those whose pockets are deep enough to afford the Big Vendors and their Big Tools.

I think that's unfortunate. I believe these circumstances have hindered (or at least slowed) the widespread adoption of ARM. ARM has virtually zero "mindshare" among the typical "corporate" developers who stand to profit the most from it. I've never met another developer, in the flesh, who knew what ARM was, let alone used it. I believe a big part of the reason why that's so is that it is virtually impossible to obtain a "working copy".

You can't just download the interfaces (or the C headers, for that matter) from the Open Group's Web site and "take it for a spin". Without a reference implementation to play around

with, the barrier to entry for the curious hacker is simply too high. People are just too busy to spend the day or so it takes to hack together a simple implementation of the ARM interfaces, just so that they can *begin* to evaluate it for their needs.

Recently, I found myself in a situation where it was abundantly clear to me that the application I was working on could profit enormously from being instrumented with ARM. I wasn't the only one to recognize the need, within our project -- it was plainly obvious. Thus, several other clever, well-meaning developers in our team had already done a good bit of work re-inventing that wheel, with a few customisations for log4j. Using these home-brewed tools, they had already begun to instrument the application, using the custom logger to trace the beginning and end of key method calls.

But their efforts were haphazardly applied, and inconsistently executed. Apart from that, there was a growing awareness that other parties had an interest in similar information -- operations, for example. Because of that, parallel efforts had begun, to try and "mine" the existing log files for similar information with Perl scipts, which then massaged the data, to present it in a Web portal.

These various, disconnected efforts needed to be coordinated, and the opportunity to use a standard, both in terms of interface and data format, was screaming to be noticed. So I suggested that we should instrument the application using ARM.

But how? Not havig the budget of a Fortune X000 or a dot boom startup company available to us, how was my project supposed to go about using ARM? Without the deep pockets of such a company, and therefore no chance of buying one of the Big Tools, how could we nevertheless use ARM? Maybe there was just no alternative to ad-hoc, haphazard, half-baked, incompatible solutions?

OpenArm is my answer to this dilema -- I decided to try to end the Big Vendor's monopoly on ARM implementations.

I don't have any illusions about the scope of OpenArm -- competing directly with the Big Vendors is not a goal. However, it isn't my intention that OpenArm be a toy application, either. OpenArm will attempt to be a full-fledged, useful implementation of the ARM standard. Only time will tell if we succeed. It's certainly worth a try.