Ecosystem Health as a Reason for Migration
The Mainframe Case

Vadim Zaytsev
Raincode Labs
Brussels, Belgium
vadim@grammarware.net

Mainframe is one of the most solid, mature, reliable... and expensive platforms, initially introduced in 1952 and currently available for a wide variety of customers. The ecosystem of the mainframe covers programming languages like COBOL and PL/I, proprietary fourth generation languages like PACBASE or COOL:GEN, transaction managers like CICS, IMS or TPF, database management systems like DB2 and IDMS, scripting languages like JCL and REXX, and a macro-assembler. Typical mainframe customers are banks, governments, travel and logistic agencies, insurance companies, as well as other big entities with large quantities of data and transaction throughput demands. It is remarkable that, despite its extreme reliability and other valuable time-proven properties, there is a tendency among many companies to migrate away from it to more mundane platforms/ecosystems. **Raincode Labs** is a company successful in this line of business for almost three decades, being involved in numerous migration projects.

A typical mainframe with 4000 MIPS costs 6–16 million dollars per year to maintain. However, this is often not the main and definitely not the only reason for migration. The way the world has changed over the last couple of decades, made it so that the companies' demands to process transactions and perform underlying queries, are growing at an increasing rate while at the same time not being linked to any financial gain. For example, in the pre-mobile era having twice as many transactions per day would mean having twice as many clients using your services, or your regular clients using your services twice as actively, or something similar, which is only positive in the long run. Nowadays it can mean the same clients being active in their mobile apps with unprofitable actions like checking and continually refreshing the status of their account or delivery. (A natural response to this would be caching the values being queried too often, but that results in a noticeable decrease in the quality of service).

Professionals that are familiar with technologies of the mainframe, are scarce and expensive. Junior developers can be educated, but are often unwilling, since it limits their future employment prospects. Being an expert in a technology or a language that is perceived as legacy, is personally profitable but somehow carries a uncool stigma. We see many of our customers deciding to migrate to an active and *cool* ecosystem just in order to be able to afford new hires in sufficient quantities. There are two more reasons intimately linked to the hiring problem. First is that using a clunky legacy system to produce software almost automatically means using clunky legacy development support tools, which is not only less attractive to developers, but also less efficient for them than relying on modern advanced interactive IDEs. The other reason is that operating within the mainframe and exhausting your technical limits means either having to purchase expensive commercial packages to cater for new or extended needs, or having to write such components yourself, both options being very costly in their own ways. Having the same problem in the Java ecosystem would mean relying on much cheaper and much more numerous third party libraries or even tapping into the vast arsenal of open source software.

Certain ways of dealing with problems on the mainframe, have consequences for the ecosystem that go way beyond the traditionally understood and researched technical consequences. As an example, consider how the recompilation is done on, say, the .NET Framework: each developer has their own piece of the overall company portfolio checked out on their laptop, where it can be compiled at any time without any consequence for the production servers. On the mainframe, the compilation has to happen on the same mainframe that handles clients' transactions. This means that global recompilation of all available sources can be harmful even when desirable, and the workload will have to be balanced and distributed over the course of days and even weeks. This in turn means that such prophylactic portfolio recompilations are significant company-wide events that happen very rarely (usually forced by new compiler releases and similar triggers). As a final consequence, it is unfortunately way too common to encounter systems actively being used in production, without their corresponding source code (since the last compilation of that system happened 30+ years ago). Even worse than that, it is possible to have several versions of the source code with different dates and sources, without the crucial accompanying knowledge of which one of them is the one currently deployed.

At the SoHeal'19, we would like to discuss software health at the ecosystem level as a reason of migration from that ecosystem even if its technical aspects are more than satisfactory, and to share our experience in migrating from mainframe to .NET, LLVM, etc. The author's short biography is available at http://grammarware.net/is. Many details of the harsh reality of mainframe migrations, are regularly shared at https://www.raincode.com/blog.