



Exploring COBOL to Java Migration

White Paper

by

Mike Kay

ESI Services, LLC

Introduction:

This paper attempts to clarify some of the issues involved when contemplating the automated migration of COBOL code to Java. Rather than arriving at a specific recommendation, because the outcomes can vary so greatly depending on a number of inputs into the assessment, the value here is in better understanding what the possibilities and limitations are, and thus being able to make a more informed decision as to whether it's a viable possibility in a particular business case.

What is the business problem to be solved?

Many organizations have an inventory of millions of lines of COBOL code, often comprising some of their most critical processes and containing business rules which have been developed over decades. Why is this a problem?

- COBOL programmers have become more rare and therefore more costly
- The COBOL systems themselves are often on expensive legacy platforms, which are growing more expensive to maintain by the year
- Organizations want to standardize technology, including programming language, interface style, reporting mechanism etc. and these legacy applications are in their own silo, all but inaccessible via these other standard technologies.

Options:

To understand the value of Java migration as a possible alternative, it's best to look at it in comparison with the other options. The possible means of improving the situation can be categorized as follows: **modernizing** the COBOL source code that you've got, **rewriting** the COBOL applications in another language, or using automation to **migrate** the COBOL applications to another language.

1. Modernize

The first option involves updating the version of COBOL (e.g., from COBOL74 to COBOL85) used and integrates the standardized coding format and constructs provided in the newer version to provide more standardized, more easily maintained code.

Advantages:

- This is the least potentially destabilizing approach – your code base is kept as is, with no risk to the business logic
- Initially, this is the least costly route
- Allows programmers to use more modern COBOL constructs which reduces the variations of COBOL that are maintained in your organization

Disadvantages:

- Your business continues to bear the growing risk and cost of the system – increasingly expensive resources are required to keep it running
- The system remains un-integrated: difficult to consolidate information, share data, utilize common UI, etc.
- More steps and moving parts are required to integrate a legacy system into an overall IT framework, such as additional communications components, data synchronization into a warehouse, and screen-scraping to name a few. This situation results in more potential for data mismatches, a greater likelihood of downtime, and greater expense associated with maintaining these additional elements.

2. Rewrite:

The second option requires rewriting the entire application suite in a different language (e.g., Java) to take advantage of capabilities provided by the new language and exploit the availability of programmers experienced in the language.

Advantages:

- This is the cleanest approach – a new, customized code base is developed that transfers the business logic to a more “modern” language
- Purpose-written code, crafted explicitly in the new toolset / language,
- Best method from a maintenance standpoint, as the new code can be written to utilize various development tools, existing libraries, and other standardized development mechanisms

Disadvantages:

- Very expensive due to the labor-intensiveness of the development effort
- Slowest approach

- Can disrupt business as processes need to be deciphered from code – a often subject matter experts (SMEs) have to interpret and/or recreate the business rules which need to be included

3. Migration:

The third option is to use automated migration technology, such as that provided by ESI and MSS, to take the entire inventory of existing COBOL source code and generate the equivalent code in Java.

Advantages:

- Much quicker than a rewrite
- Preserves existing investment in business logic
- Achieves the objective of freedom from COBOL, as well as potentially from the existing platform / environment
- Code can now be maintained using the current toolkit

Disadvantages:

- Migrated code is not optimized, and also is typically much more difficult to maintain than code that was originated in a given language
- Specialized functions in previous applications may need to be migrated manually
- Staff will need to be educated in new programming and database technologies if not already in use at your organization

Questions:

Do you have a mandate to be on a new technology or off of your existing technology? If so, is there a stated time period to complete the change?

Are you having difficulty finding and/or keeping qualified COBOL programming personnel?

Have you found commercial packages written in other languages that you would like to integrate your COBOL applications with but are unable to?

Conclusion:

If you are in a situation that requires you to move to a more current, portable technology such as Java, further investigation into automated migration technologies may very well be the approach that best suits your technical, timeframe, and cost parameters.