When the decision has been made to modernize and migrate off of the mainframe, there are a number of viable paths. They range from a simple COBOL-to-COBOL Re-Platform to a total Re-Architect and Rewrite. In the middle are variations of an automated conversion. The following list summarizes the options:

- COBOL-to-COBOL Re-Platform
- Automated Migration/Conversion
  - High Yield Syntactic Translation
  - Low Yield Semantic Transformation
- Re-Architect and Rewrite

Let’s examine the advantages and disadvantages of each option in more detail.

**COBOL-to-COBOL Re-Platform**

This option is a proven migration path and results in getting off the mainframe with the least amount of disruption. However, it does little to move the application into the modern object-oriented application development world. In addition, the enterprise is still dependent on the ever dwindling population of COBOL programmers.

**Re-Architect and Rewrite**

The clean-slate Re-Architect and Rewrite approach offers the most benefit but comes with the highest cost and risk. This path is initially favored by most enterprises because it offers the perfect outcome, a totally new and modern application. However, usually in the planning phase, it becomes clear that there is a large volume of application functionality that must be replaced and that the internal workings of the code are not documented and many times not well understood by the current staff. This increases the cost and risk of this approach.

**Automated Migration/Conversion**

There are essentially two types of automated migration and conversions. Both types attempt to automate the conversion process with varying degrees of success. Let’s look at each option:
High Yield Syntactic Translation

This approach is a highly automated translation of the application into the new environment. It is based on syntactic translation of each statement into functionally equivalent code. The result is a near perfect automated translation which runs identically to the original application. In some ways it’s similar to the COBOL Re-Platform option except that the application is now running in a modern development environment.

On the negative side, the converted application generally does not look like a clean-slate design. It carries the basic logic of the original COBOL program and usually is dependent on some proprietary run-time support code.

Low Yield Semantic Transformation

The semantic transformation attempts to solve the drawbacks of the syntactic translation by dividing the semantic meaning of the COBOL code and writing a new application to implement that functionality. Due to the challenging nature of this task, this is a low yield process requiring significant manual intervention and coding in order to produce a functioning application.

The main problem with this approach is that the amount of cost and time spent on coding and debugging the converted application frequently approaches that of the Re-Architect and Rewrite path. Furthermore, the quality of a rewrite is always much better than the automated output.

Recommendations

If time and budget is not a concern, look into Re-Architecting and Rewriting the entire application. If resources are limited, only rewrite the core application functions, using High Yield Syntactic Translation for the rest of the application. If the time factor is critical, do a fast High Yield Syntactic Translation of the entire application, then when off the mainframe, rewrite core functions.

Avoid the COBOL-to-COBOL Re-Platform option because this does nothing to move the application into a modern development environment and keeps the application dependent on a shrinking pool of COBOL expertise.

Rather than using a Low Yield Semantic Translation, opt for a Re-Architect and Rewrite. It will produce better results with only slightly higher costs.