



Databorough

Automated RPG Reengineering Save Your AS400 and Your Job

Steve Kilner

RPG developers have wondered for years if there would ever be a way to automatically convert those giant, old, interactive RPG code monsters into a modern architecture.

*Is there any way to salvage all that code that has so much vital, valuable business logic in it? And make it **more maintainable**? And truly **modern**?!?!*

Rather unbelievably, a solution has emerged. And it came from a direction maybe no one was expecting. But is totally logical once you think about it.

As many people know, for 25 years Databorough has been continuously developing and enhancing its X-Analysis product for analyzing and documenting RPG, COBOL and Synon code.

Additionally, there has been a long line of specific use reengineering products: Y2K, field expansion, SQL and Unicode conversions and so on.

As someone who has worked on the internal code of X-Analysis, I can tell you that there is *an amazing amount of intelligence and information* in its analytical programs and the repository it creates.

To Restructure a Monster RPG Program You Need a Monstrous Amount of Information

In the last couple of years it has become apparent that the repository had reached a critical mass of information and analytical capability, and there was the means to *finally solve that RPG restructuring puzzle*.

Here are a few samples of the knowledge a repository needs to contain in order to feed an automated restructuring process:

- All the possible screen flows in a program
- The events that trigger each screen flow
- The statements to execute before displaying a screen, no matter where they are in the program
- The statements to execute after displaying a screen – validation, output, etc.
- The different options that users may enter into subfile selections, and what code they execute
- All the fields that feed into a given field - which fields are updated with values from database fields

Over the years Databorough has built an incredible analytical machine. The X-Analysis repository of one large scale ERP application consists of:

25,107 objects
13,391,232 lines of code
205,261 pgm to pgm relationships
77,292 pgm to file relationships
489,764 business rules
124,519 fields
19,274,851 field and pgm variable instances

This is just some of the information that enables automated, intelligent restructuring of monolithic RPG code.

Trust me, there are hundreds of such knowledge elements, and what Databorough realized is that they had most of them in their repository.

If you've ever done any programming of in depth source analysis, one thing you

learn is that it builds on itself: the more you do, the deeper you can go.

Databorough hit the point where they had all the information needed to undertake the seemingly impossible task of restructuring monolithic legacy RPG code.

What Does Restructured RPG Code Look Like?

Most of you have probably seen articles or watched presentations about modern architectures. Going back to the 90's you heard about *object oriented architecture*, event-driven designs, model-view-controller patterns, and a little later *REST, a web-oriented, stateless architecture*.

The goals of all such modern architectures are:

- Modularity
- Loose coupling
- Reusability
- Distributability
- Scalability
- Maintainability

A crucial goal for legacy code modernization

Make it more

A key strategy

Restructure your code before you convert it.

One of the insights that Databorough had was that an optimal tool would first convert old RPG, not to Java, C#, or something else, but rather would first convert to RPG itself.

And convert it to *RPGLE/Free, fully modularized as exportable procedures* with the equivalent of a session bean, in order to enable a stateless, event-driven, MVC pattern.

An RPG-to-RPG conversion had the great advantage of being fully understandable by the same developers who understand the existing code.

For more detailed information and a diagrammatic explanation of how monolithic RPG code is restructured, see the [Visual Guide to Automated Reengineering](#).

If Converting to Java or C#, First Restructure Your RPG

Over the years, a number of organizations have tried to simply convert code in ways that essentially amount to line by line conversion. Question: where are the success stories?

Indeed, Databorough followed this same path and tried to make it work. It wasn't until the product developers themselves, out of frustration, asked if they could first restructure the RPG that it became clear that this was the only possible successful strategy.

Seeing the same tangled mess of code with unexposed functionality in the same old monolithic, procedural architecture is actually a step backwards, because now *no one can understand it*.

Lesson Learned

Do not convert to another language without first restructuring program architecture. Otherwise:

- you will have code that *no one understands*.
- you will not achieve the goal of making it *more maintainable*.

And think again of the goals of modern architectures listed above – *would a line by line code conversion achieve any of those goals?* Of course not. That's why the only answer for language conversions is to first restructure your code into a modern architecture.

Once restructured, converting RPG code into a modern language is natural and organic. Most importantly, it is traceable and understandable. Not only can the restructured RPG be traced into an appropriately structured modern language, but Databorough's conversion tool features a couple of key facilities: 1) side-by-side code comparisons and 2) generated pseudocode, again with side-by-side comparisons.

Restructuring: The #1 Strategy For All Legacy AS/400 Organizations

What Databorough realized was that whether an IT organization planned to stay with RPG and the IBM i, or move away from it, by first restructuring to RPG, and bringing modern architectural structure to old monolithic code, it *opened new doors for organizations struggling with aging, increasingly costly and complex applications.*

Restructuring is an important tool to extend the lifetime of RPG applications and continue to utilize staff skills.

Converting old monolithic code to a modern structure brings a variety of benefits to the application:

- It is more comprehensible and maintainable
- It is now prepared to be deployed as SOA
- Its useful life has been extended for years because it is more maintainable
- Existing staff skills and experience can continue to be leveraged
- If IT has the direction of moving to other languages, the restructured code is now intelligently convertible to Java or C# in a future-oriented, maintainable way

If Your Strategy Is To Move To Java or C#

Restructuring first to RPG is the best way to convert to Java or C# in a way that is intelligent and coherent while still using modern patterns & practices.

Databorough is offering a free trial to interested users who would like see their own RPG code restructured into a modern, object-oriented, event-driven, stateless architecture.

To sign up for a free trial, or to see more detailed information and a diagrammatic explanation of how monolithic RPG code is restructured, see the [Visual Guide to Automated Reengineering.](#)