

Source: Article in Midrange Magazin January 2008:

Migration of System i RPG Applications

Enterprise Generation Language

With the Enterprise Generation Language (EGL), IBM offers a modern development environment in which both System i native applications and platform-independent Java apps can be created. A lot of what you missed in RPG is now comfortably available: from web services to web-based user interfaces, from a drag & drop GUI to highly productive assistants, from database modeling to total platform independence.

Human and application evolution

Many companies around the world continue to trust in System i applications and the RPG programming language when it comes to critical business applications. Even today, RPG applications from the early days are still running unchanged. Yet over time, existing applications have continued to grow more comprehensive and complex. Because the existing programs have always been executable in new versions of i5/OS without having to make any changes, there are still a number of programs around today that were developed with technology that was state-of-the-art 15 or 20 years ago. There are also programs that use the latest RPG IV features and a seemingly endless range of variations in between.

Parallel to System i and RPG, new development platforms with modern graphical user interfaces have been created like Java and .NET. Because these platforms have now established themselves as standards, a real generational change has become apparent in application development. In order to stay on top of things, RPG developers and the applications themselves have to master this generational change sooner or later.

Yet past investments made in the existing applications and staff skills have usually been quite costly, which is why this generational change can only be accomplished in an economically feasible manner if a large proportion of the existing apps and skills can be sensibly transferred to the new platform and serve as a launching pad for step-by-step modernization. The migration of System i RPG applications is therefore a key to this generational change.

When does it make sense to migrate RPG applications?

There are a number of triggering factors for migration, which either individually or in the sum of their parts make migration indispensable.

Expected remaining lifecycle is greater than 5 years

Young RPG developers are a rarity today already. In five years from now, RPG will be considered an exotic programming development that nobody will want to invest any new training in. The maintenance of many critical applications will then no longer be warranted.

The application is needed for another platform

Within the scope of platform strategies or consolidations, the application is needed for a non-System i platform such as Linux, Unix, Windows or zOS. Within the scope of database strategies or consolidations, a DB2/UDB, Oracle or SQL server is needed in place of DB2/400.

Transition to a service-oriented architecture

Existing monolithic architectures have to be broken up into a business process or service-oriented architecture. Supporting the service concept through language and development environment is vital. Optimal support of web services and languages for business process modeling such as BPEL must be ensured.

Necessity of graphical user interfaces

Users who mainly work with Windows or web apps find the transition to the green screen application interfering with the operational flow. There is no sensible integration between the interfaces and operation usually involves a high degree of extra training. In addition, management often sees green screen applications as an indicator that the IT department is not really that innovative.

Agility improvement

It is very difficult to modify applications that have grown over time. The introduction of new business processes can thus become quite expensive and be considerably delayed. The company's power to compete is therefore seriously encumbered.

Boosting competitiveness

Vendors of RPG standard software in particular have huge problems gaining new customers, who prefer to invest in Java and .NET applications.

RPG developer skills getting scarce

A larger number of experienced RPG developers will be going into retirement over the next few years. Apart from weakened innovative power, the maintenance and development of existing apps is endangered.

Consolidation of development teams

Nowadays many companies have mixed developer teams for RPG, Java and perhaps even older 4GL applications. This makes development less flexible and more costly. Merging different development teams into one standard modern language and development environment increases flexibility and cuts costs.

Migration – to some extent in conjunction with reengineering and modularization – is the foundation of the solution to these problems. However, for migration to make sense, the app to be migrated must exhibit a certain minimum standard of quality.

Protecting investments

Traditionally speaking, there three different approaches to a generational change in application development:

Replacement with standard software

The existing application is replaced by standard software from the respective vendor. This takes a lot of effort because it is tied to organizational changes. It also creates new dependencies on a vendor. It may make sense for standard processes, but not for competitive-critical processes because the differentiation potential is highly limited.

Redevelopment

This is tied to high costs and scheduling risks. Existing RPG applications are often not documented in enough detail and their functional features are therefore not given enough consideration during redevelopment. When introducing the rewritten functionality, this can lead to considerable interferences in the flow of operations.

Migration

In an automated procedure, the functionality of existing software is completely or incrementally transferred to a modern development environment. Risks and costs are greatly reduced. In the course of migration, quality improvements can be made by reengineering or the structure can be modularized.

Why migrate to EGL?

Aside from Java and .NET, IBM's Enterprise Generation Language EGL is particularly suitable as a modern target environment for critical business applications. EGL combines the strengths of Java with high performance native code engineering for System i and System z. The applications therefore run optimally on all important platforms from Windows, Linux, or Unix, to System i and System z.

Embedded in Eclipse, EGL offers a modern, service-oriented language for the efficient development of different application types like web programs, database apps, web services, and batch and high performance servers for fast transaction processing.

Thanks to the tightly knit integration in the Rational tool chain, EGL development can be embedded in a professional development process or lifecycle, starting from requirements management to modeling and coding, all the way up to testing and deployment.

To allow for the optimal migration of RPG applications to EGL, the migration specialist PKS Software works closely with IBM's EGL development lab in Raleigh. In addition to facilitating the migration of RPG apps, EGL is also easy to adjust to, allowing for an efficient transition of RPG developers and their important business know-how in just a fraction of the time it would take for Java.

What migration scenarios are available?

The application remains on System i

If the migrated application is only needed on System i, this is very simple because the existing RPG programs can be migrated to EGL step-by-step. i5OS-specific functionalities like CL, commands, APIs and system commands are often used in the applications. After migration, they are all still available, thus making replacement unnecessary.

Migration is typically done in two phases. First off, the existing RPG app is decoupled from the 5250 user interface using a server builder tool. The RPG application now works with the API functions which in turn can directly access a web-based or Windows user interface. This GUI can be automatically created from the existing DDS source through a GUI stylesheet with over 60 new functions. With very little effort, the application is then transformed to a server application that features a graphical user interface and no

longer requires any interactive performance.

In the second phase of migration, the RPG programs are successively migrated to EGL. The generated EGL programs can use the same API for the user interface as the RPG programs. New functionality can now easily be added in EGL thanks to the simple integration with RPG and the graphical user interface. New EGL programs can be developed directly within a service-oriented architecture. Over time, programs in need of improvement can be recreated in EGL with this method. By refactoring, the existing architecture can also be transferred to a service-oriented architecture.

The result of migration: all platform-specific functionalities and database access from RPG programs are encapsulated in EGL libraries. By separating business logic and the platform-specific layer, the applications' portability is significantly increased.

The application is needed for Windows, Unix or Linux

Principally speaking, because EGL is platform-independent, migrated RPG programs can also be used on other platforms. However, due to the comprehensive functionality of i5OS, some specific features have to be considered. In Windows, Unix or Linux, there is no standard database like there is in i5OS, which is why a database such as DB2/UDB, Oracle or MS SQL server is also needed. Because RPG applications prefer to use functionalities from i5OS that are not available in Windows, Unix or Linux, these have to be extracted from the application and then made available in a special service library. Extraction can be either be done manually, or semiautomatically through rule-based reengineering. Since there is no RPG compiler available on these platforms, the application has to be completely migrated to EGL in order to be executable. With the corresponding tools, existing data from DB2/400 can automatically be transferred to DB2/UDB, Oracle or a MS SQL server.

The application is needed for System z

If desired, existing System i RPG applications can also be migrated to System z with EGL. There are two methods to do this, but they greatly differ in their complexity.

The first approach consists of migrating the DB2/400 database to DB2 in zOS. All RPG batch programs are also migrated to EGL and are very powerful in zOS Cobol. The job control language CL from System i is either manually or semiautomatically migrated to JCL via rule-based systems. Interactive RPG programs can be migrated via EGL to zLinux and they also work with DB2 on zOS. A web interface is generated as the user interface. The interactive EGL programs then run in Java and zLinux.

The second method foresees migrating all interactive programs via EGL to zOS in addition to the batch programs. To make the interactive programs CICS capable, overall reengineering of the RPG/EGL programs is necessary so that a switch can be made from procedural architecture to transaction-oriented architecture. This can either be done manually or, again, semiautomatically through rule-based systems. The result is a purely zOS application.

Migration tools

There are a number of tools and libraries that make migration possible:

Migration tools 400 EGL

- RPG to EGL converter
- Database scheme generator
- Database loader
- Universal Client Xi
- Server Builder 400
- Service Library 400

Migration example

The following screenshot shows an RPG program (on the left) and the EGL program generated from it (on the right). For RPG developers, the EGL program is quite easy to understand.

