Evaluation of ARCAD MaaS Conversion Process from SYNON to Modern RPG

Jim Diephuis
diephuis@us.ibm.com

IBM Lab Services Power Systems Delivery Practice

Executive Overview

Introduction

Lab services was asked to evaluate the maintainability of the code that results from the ARCAD Modernization as a Service (MaaS) process that takes RPG or RPG ILE code generated using SYNON methods and converts the code to modularized, free-form, modern RPG. The result is code that can be more easily managed and maintained without the need for SYNON.

The review also included an interview with Geodis, one of the companies that has been converted from SYNON to ILE RPG using MaaS.

Two projects were reviewed. One was the demonstration project 'horse race'. The other was the code generated for Geodis. Both the original code Generated by SYNON and the final code generated by the ARCAD SYNON transformer were examined.

Methodology and Summary of Findings

The evaluation of the RPG code for maintainability centered around four main criteria:

- Is the code understandable?
- Is the code modifiable?
- Does the code meet requirements?
- How easy is the code to test?

The definitions for these criteria were taken from: https://quandarypeak.com/2015/02/measuring-software-maintainability/.

The following tables summarize what is required to satisfy each criterion, and what was found in the provided code.

Understandable

Criteria	Findings in ARCAD RPG Code

• Is the code easily readable?
  ○ Is it self-descriptive?
  ○ Is it documented where needed?
• Do the procedures serve one and only one purpose?
• Are the sizes of the procedures manageable?
	• Yes, the code is readable and self-descriptive.
	• Comments are enough and where needed
	• Procedures appear to serve one purpose
	• Procedure sizes are very manageable

Criteria	Findings in ARCAD RPG Code

Modifiable

Criteria	Findings in ARCAD RPG Code
• Is the code easy to follow? • Does the code avoid duplication? • Are the programs/procedures generally cohesive? • Are the procedures and programs documented sufficiently?	• Code is easy to follow • Duplication generated by SYNON has been taken out • Programs and procedures are cohesive • Documentation is enough

Requirements

Criteria	Findings in ARCAD RPG Code
Does the code meet requirements?	• The resulting code has shown to function equivalent to the SYNON generated code.

Testing

Criteria	Findings in ARCAD RPG Code
How easy is it to test changes?	• The ARCAD Verifier test automation solution was used to verify the transformed code worked the same as the original code. How easy it is to test changes in the future will depend on where and how the changes are made. The Verifier tool can continue to be used. Changes to procedures that do not require use of a user interface to test will make testing easier in the future.

Tools Used to Assist in the Evaluation

1. Rational Developer for i (RDi) tool to browse the before and after code
2. ARCAD Observer to analyze the code

Evaluator Skill Set

1. Evaluator is a Senior Managing Consultant whose focus is assisting clients with moving their code to modern RPG and help them understand modern programming methods. He regularly shows clients how to use RDi to improve their developer's productivity.

2. Evaluator is a regular speaker at IBM I's Large User Group (LUG) on the topic of application modernization

ARCAD Evaluation Sheet

Quality of the generated code?

How close is this code to resembling something that might have been hand re-written by a person?

There are fewer service programs than would exist if it were re-written by hand. This is primarily because all the SYNON functions are bound into one service program, STD_SRVPGM. While the source members are well separated with multiple procedures that deal with a specific concept, like date processing, all these functions are put into one service program. That service program can have hundreds of procedures and modules bound into it. This will make for a large amount of unneeded code that might not be used being loaded into a job. If this would be 'written by hand' there would be more service programs that would be used for specific needs, like date processing or object manipulation.

There are also cases in the condition source where multiple members had constants created that had the same value for the same type of constant. For example, in the 'horse race' example there are two source members that contain constants for gender, 'M' and 'F', one for horses and one for jockeys. This could be handled by a single source member for gender and shared by the condition code members. A similar example in the Geodis code is multiple members with a OUI and NON constant ('1' and '2') that could be shared.

That being said, with the common functions and conditions that are generated by SYNON already separated out, it will be easier to restructure the functions into proper service programs than it would be to try and figure it all out as part of a rewrite. The coding time will be greatly reduced as the procedures and prototypes are already done.

What do you like about the generated code?

The long names provided through using SYNON's repository and using aliases in the data file definitions is a plus. It makes the code more readable than staying with short names that have no meaning This is especially nice because of the many fields generated by SYNON.

Putting the final code transformation into free form and moving it to the first column makes the code more readable and understandable for developers that don't have an RPG background. This is helped by pulling out much of the repetitious code SYNON plugs in. Looking at the SYNON generated RPGLE code next to the final, converted code is like night and day for readability.

There are a lot of 'little things' that ARCAD did that makes the code more readable and manageable. The final code extracted many of the data specifications and either put them into copy books for reusability or removed them because they were only used as intermediate fields. Removing the tables that SYNON used for constants and literals simplified the code. Putting

cascading IF THEN-ELSE statements into a SELECT statement reduces the number of ENDIF statements and makes the options more understandable as well as reduces the amount of code to manage.

What do you feel could or should be improved?

It is difficult to answer what could or should be improved. While it would be nice to say it would be good to have the business logic separated from the user interface, or that data access objects should be created rather than having SQL or record level access exist in the main programs, I am not sure that is practical. Those are really 'next steps' in the modernization journey. Most of the functions and repetitive code have already been separated out, making the next steps easier to do. Determining what is user interface versus business logic is not always an easy thing and is better done in individual decisions. ARCAD has partnered with Profound UI to help clients put a web interface in place of the 5250 interface. This is a good next step for those that want to move away from 5250 interfaces. It makes sense to not do this as part of the initial process as it makes the output easier to verify that it gives the same results as the original programs.

Some of the variable names seemed too long to be practical. This is adjustable based on the client's desires. For the COBOL developer coming to RPG however, as Geodis was, long names make sense. With RDi's content assist the long names are not as difficult to deal with as they would be otherwise. It seems strange to complain about names being too long as they now have meaning rather than being some made up six-character variable name that has little to no meaning. The main reason for my concern is the opportunity for developers to misspell them and the amount of time it takes to type them. RDi's content assist will reduce this possibility.

Maintainability / style of the generated code?

Using modern, standard tools, how easy would it be for a skilled RPG developer, with some source application knowledge, to take this application on and support, maintain, re-factor and enhance it?

With Rational Developer for i and ARCAD's Observer as well as knowledge of the source application, a skilled developer should be able to support, maintain, refactor, and enhance the application easily.

What about this code contributes to its ease of maintainability?

By breaking the code out into five different source members, instead of having just one or two source members, there are fewer members to struggle through to find the ones wanted. While there are still many members that is to be expected in any large application.

Another key is that it is free form RPG with the code appropriately indented and understandable names. This makes it readable and easier to comprehend than fixed form RPG. Also, by taking out

common functions, there is less code getting in the way of understanding what the programs are doing.

What do you feel could or should be improved?

Constant values could be all upper case. They do start with a 'c_' which helps identify them, but it is somewhat of a standard that constants be capitalized. ARCAD is looking at this as something that they will integrate into the process soon.

It is surprising, considering the many members containing constant names in the conditional source file, QRPGCNDSRC, that common constants like 'Y' and 'N' were not given constant names. Having a set of constants that are used throughout the code for common values would improve readability as well as isolate where changes might need to be made. Using *BLANK instead of ' ' should be easy to do when managing constants. Along the same line, indicators could be set using '*ON' and '*OFF' rather than '1' and '0'.

There are converted source members that were defined as SQLRPGLE but did not contain any real SQL. One statement was put in as a dummy statement to make it have SQL. While the original version had SQL, all of that has been moved elsewhere. This is for the user exit programs. There is no reason to continue to create these as SQL source members.

The architecture of the transformed application

How viable and modern do you consider the new architecture to be?

This really isn't a new architecture. It is a transformed architecture from SYNON to modern RPG. The look and feel to the end user are no different. The difference is that the development team is no longer constrained by the limitations of SYNON. For instance, Geodis took their 5250 application and, with the help of Profound UI, converted it to have a web-based look and feel.

Many of the user exit functions that clients wrote to work outside of SYNON can now be integrated into the code. This should make the application easier to modify and enhance.

While SYNON was a great solution for quickly creating and enhancing an application in the 90's it did not keep up with the changes happening in the industry. Converting the SYNON code to modern RPG, along with cleaning up and modularizing the code, will allow companies to move forward with further modularization and development of new features without being tied to generated code. Clients could just take the SYNON generated code but then they would have still been stuck in old models with short names and replicated code. Even just running the code through ARCAD's RPG transformer to get free form RPG would not have helped that much. The extra effort that ARCAD added to take out the repetitive functions and use SYNON's name repository to give fields long, descriptive names is what makes the final product viable as a modern architecture moving forward.

What do you feel could or should be improved?

This is a great first step in getting a client off SYNON and into an environment where they can start to take the next steps of modernizing the code to allow for multiple user interfaces and using modern database techniques. By doing Modernization as a Service ARCAD allows the user to start taking these steps as part of the service or wait and do it later. As such it is hard to find something that could be improved with the approach or the final architecture.

It is a little limiting with the code being generated as RPG while many SYNON users are familiar with generated COBOL code. It is understandable as RPG is the predominant language on the IBM i and modern RPG is more adaptable than COBOL for making modifications. Ultimately, RPG is the correct language to use. With ILE, clients will still be able to use COBOL for user programs if they wish to. It could be a stumbling block for some clients that wish to stay a COBOL only shop.

Application documentation (Comments, annotations, etc.)

Well-placed comments, annotations and other 'documentation' type objects can greatly assist in code comprehension, maintainability and readability. Did these objects come over properly?

The comments in the code helped to make the code understandable. Many of the comments were there from SYNON already. Headings were added to help developers understand where code came from. This will be a help in the short term as developers that knew the application in SYNON will be able to relate to it. Over time, as enhancements are made and new developers come on board those comments will be irrelevant.

Having a tool like ARCAD Observer is how developers really get to understand the application architecture. While the end user has a choice as to whether to continue to use it, it is recommended to have some tool like Observer for developers managing a large application.

Do they help or hinder? What do you feel could or should be improved?

More comments are always welcome but only if they add to the code. It would be difficult to add more comments without understanding more about the application itself, which is not the function of the transformation.

Would you use such a solution and / or recommend it to others to use?

This is a great step for moving people off an antiquated SYNON tool that has been stifling the growth of many IBM i clients for many years. This process is recommended to users still stuck on SYNON that need to modernize their application.

An interview with a client

ARCAD has already converted a few clients using the MaaS process. We interviewed Hugues

Clement the IT manager for GEODIS, to see what they thought of the resulting code and what their experience has been as they move from SYNON to modern RPG. GEODIS was the first client that ARCAD converted. GEODIS had been using SYNON COBOL for their key application. They were finding that SYNON was restricting their ability to take advantage of many of the modern database features available on the IBM i. It was also restricting their ability to respond quickly to their needs. They went from SYNON with COBOL to free-form RPG as well as moved from record level access to using SQL. They also used Profound UI to move from 5250 emulation to a web front end. ARCAD spent time training the development team on RPG and Rational Developer for i (RDi).

The comments below are paraphrases of Hugues' responses.

What has been your experience so far with making changes to the code?

Now that the development team is comfortable with RPG making changes is easier than it was with SYNON. The free form RPG code is easier to comprehend than the fixed form RPG generated by SYNON.

What new features have you added since the conversion?

There are several new projects involving more use of SQL and providing web services. These projects would have been very difficult, if even possible with SYNON.

How well have your developers adapted to the code?

It has taken some time to get comfortable, but the developers are liking it. There are many features of free form RPG that make the code easier to write and understand.

Have you found the code to be easy to maintain and manage?

Yes

What changes would you have made to the final product to make things easier for you and your team?

There is nothing with the output of the tool that would change. More training integrated with the modernization might have been useful.

GENERAL COMMENTS/Summary

In general, the process ARCAD is using to help SYNON clients move off the tool is very impressive. It gives the clients many options depending on where they want to end up. This has been a sticking point for many SYNON clients on IBM i for many years. While not a perfect solution or the end of the modernization journey it does give them a good start on that path. The number of options that ARCAD provides the user with how to transform the code and even the database is

great.

The decision to do the transformation as a service is the right one. Just providing people a tool with no additional assistance would have caused frustration. Also, being able to do most of the work on ARCAD systems so the bulk of the work could be done without impacting day to day operations is a big plus. Only having to freeze code changes for a couple of weeks versus months is also a big plus.