COBOL V5 Migration Strategies

Session 17033

Jim Liebert
March, 2015

Contents

Introduction

COBOL V5 represents a dramatic leap forward in the evolution of COBOL. Very nearly a complete rewrite of the language, COBOL V5 moves from a classic compiler to the code generation model used by Java. This re-architecture offers significant performance improvements, both by implementing aggressive optimization enhancements and by leveraging new hardware instructions introduced over the years. COBOL V5.2 continues this trend by quickly providing even more performance opportunities by exploiting the z13 hardware. COBOL V5 represents a win-win situation; good for IBM and good for IBM customers.

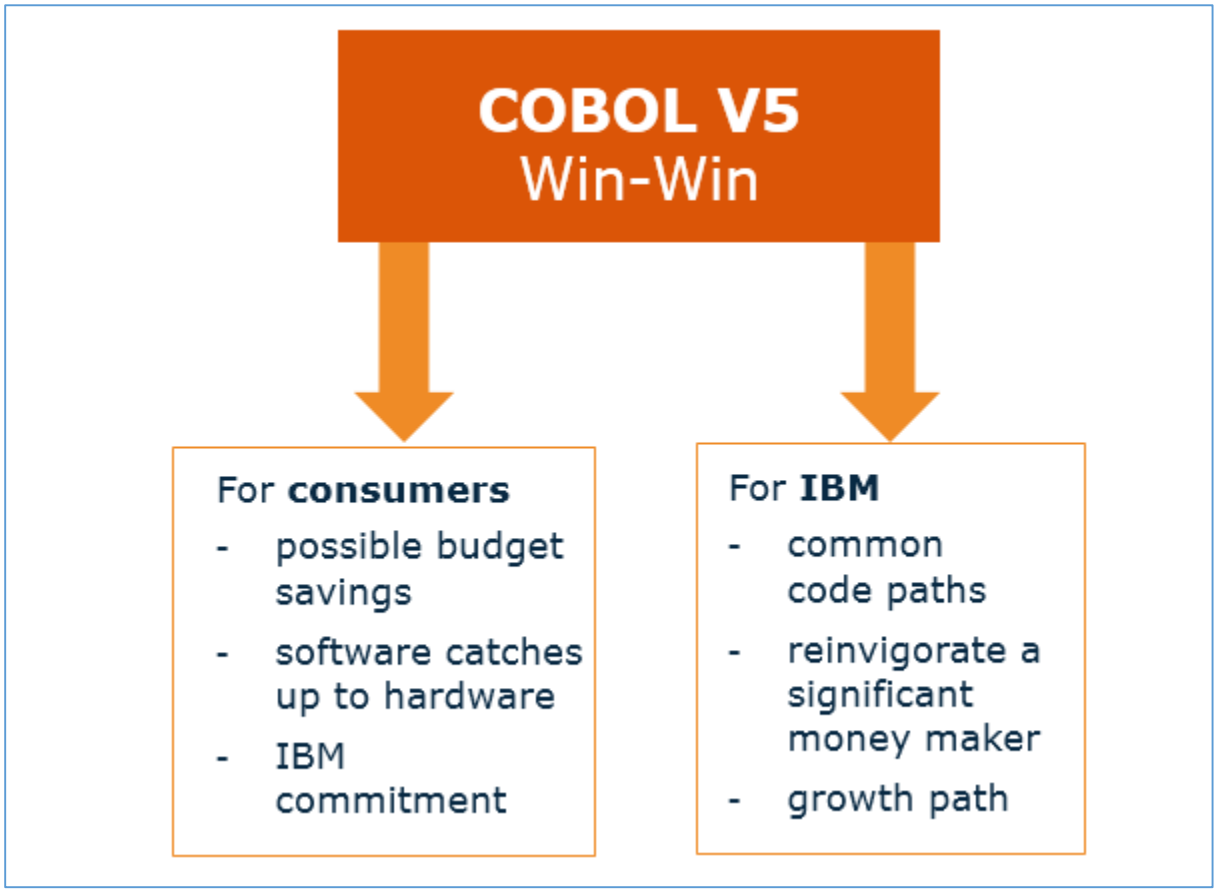

Figure 1 COBOL V5 Win-Win

However this new architecture has also introduced some migration challenges for sites that rely on COBOL for their day to day processing. There is certain COBOL syntax that is no longer supported, the support system around COBOL has been changed and certain compile decisions must now be revisited. The changes are dramatic enough that a site must also commit to a certain amount of regression testing to verify programs as them move from older COBOL to COBOL V5. The reward is high; the risk is fairly low; but the effort (and time) is not trivial. This strikes at the very core of the DevOps mantro – "how much time implement a one line coding change?"

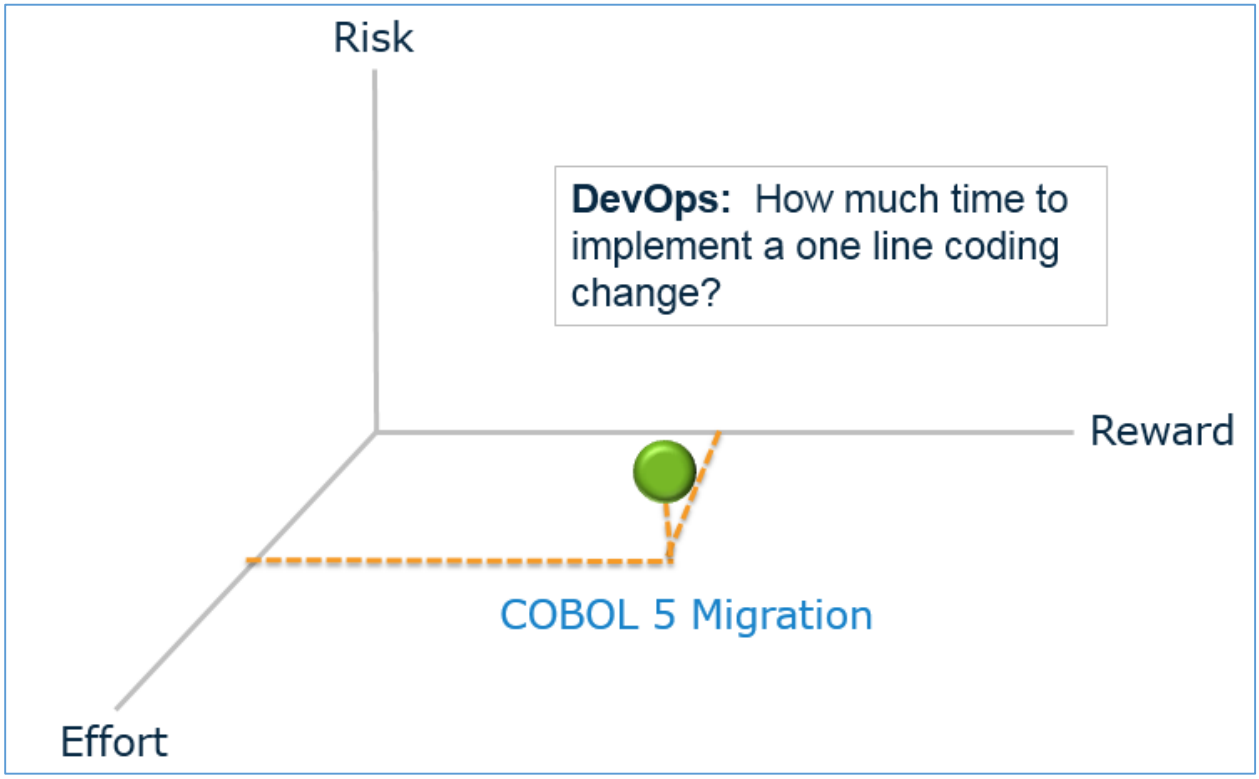

Figure 2 COBOL V5: Risk, Reward, Effort

This handout is based on the Share presentation 17033 given at Share in Seattle 2015. It offers some suggestions on how best to accomplish this migration to COBOL V5 at your site – to realize maximum benefit while also making upper management aware of the effort involved in the migration.

Preliminary Steps

The first step is to complete some preliminary research and make a plan. We recommend reading the COBOL V5 Migration Guide to best understand the technical changes that will be required. A nice summary of the Migration Guide is available in Tom Ross' handout for Share Seattle presentation 16615. The most current Migration Guide is available in the Enterprise COBOL for z/OS documentation library.

The second step is not mandatory but is recommended: start an official project within your organization. Although migrating to a new version of a programming language usually falls into technical debt and is handled in the course of general development work, this migration is significant enough to warrant a project, it will require significant time investment by many groups within your organization and offers significant performance improvements that should be tracked – and measured to determine any associated dollar savings.

Both these steps and the first phase steps discussed in the next section could be completed prior to ordering COBOL V5. The timing of the order is important in that IBM offers only a limited grace period of running multiple COBOL versions at a site before they start charging. The goal should be to complete your migration prior to the end of that grace period or to petition IBM to extend that grace period. Paying for two versions of COBOL is an expensive proposition.

Phase 1

Phase 1 involves preparing your system for the COBOL V5 migration.

Currency

COBOL V5 has certain subsystem requirements. As of this writing, these requirements are not onerous – you just need to be fairly current.

Prerequisite levels of related software products

- z/OS V1R13 or later
- CICS Transaction Server for z/OS, V3 or later
- IBM DB2 V9 or later
- IBM IMS V11 or later

Figure 3 COBOL V5.2 requirements

Complete LE runtime migration

Hopefully this work has already been completed. This is best covered in the IBM manual Language Environment Run-Time Application Migration Guide. This step can be summarized as ensuring you have removed any reference to old COBOL run time libraries (such as COBLIB or COB2LIB) from JCL or from your LNKLST or LPALST. Those references should be replaced with the associated LE libraries such as SCEERUN.

Convert Load libraries to PDSE

COBOL V5 executables are Program Objects and can only reside in PDSE datasets. The quickest way to resolve this migration step is to convert your user group load libraries to PDSE format. This is a mostly mechanical step but does require some planning and a conversion strategy. While this step could also be accomplished piecemeal as it comes up in Phase 2, accomplishing this task prior to ordering COBOL would speed up the process.

SCM drives all compiles

This is also a step that you may have already accomplished. It is important that your SCM promotion process drives all compiles. If programmers are controlling compiles via JCL, it will be difficult to guarantee they are using the desired compile options at the specific promotion points.

Your SCM administrator will also have to give some thought as to how aggressive your organization is going to pursue performance gains. For instance OPT(2) is so aggressive as to obviate the ability to debug a program. Your organization will have to decide if, when moving a program to production, performance or debug ability is the overriding goal. The recommendation here is to compile with OPT(0) up to moving a program to production. At the promotion to production recompile at OPT(2). Of course your organization will have to weigh the advantages and disadvantage to this strategy.

Another decision point for the SCM administrator is the setting of the ARCH compile option. Tailoring it to match your hardware (for instance ARCH(11) if you are running a z13 box) will achieve the greatest performance gains. Again, these will have to be weighed against your environment specifics: can this code be run on various systems with differing hardware? Are there DR limitations to take into account? The recommendation here is to tailor your ARCH setting to your lowest common hardware denominator. A second suggestion may be to set up your DR environment so as to run on the same hardware your active system uses.

Finally your SCM administrator should take into account the JCL changes required by the COBOL V5 compile:

- Requires very large region size – possibly up to and beyond 200M.
- Compile will run much longer so make sure they are in an appropriate class.
- Requires JCL changes to the compile step.
- Generates allocating a larger object deck.

Order COBOL V5

At this point you will have accomplished all the prerequisites and can order the COBOL product. This step should also include getting as current as possible with COBOL, with the various subsystems and with your ISV software. Getting current will very likely avoid problems that were hit by other customers.

Implement your SCM compile changes

Once COBOL V5 is installed and functional your SCM administrator can now activate their JCL changes associated with your development promotion processes.

Phase 2

Pilot Project

Prior to releasing COBOL V5 to the general population, the recommendation is to run a quick pilot project on a minor application. This will allow you to:

1. Determine any flaws in your steps as set up in Phase 1.
2. Evaluate how much regression testing should occur to give your organization the most confidence that problems will not arise as you move the code into production.
3. Determine a rough estimate of the level of performance gains your site may experience moving forward.

Migrate programs as they come up for changes

Once the pilot program is completed, the recommendation is recommended that for a given amount of time (as determined by your organization) you only migrate programs as they come up for changes, either due to bug fixes or active development. The theory being that the program already requires changes and that this the optimum time to also migrate them. Extra considerations:

1. Will there be some exception criteria to allow specific cases to *opt out* of the migration step? For instance, when addressing priority 1 bugs?
2. Again how much regression testing to include in the mix and how much added time will your organization be willing to add in implementing even modest changes?

An alternative approach may be to migrate *applications* as programs contained in them come up for changes. This is a more aggressive approach that will cost more in time on the front end but may save time in regression testing and also increase the potential CPU savings.

These approaches could be mixed within an organization and the decision left to the various application groups, depending on their evaluation of risk, effort and benefit.

Publish Results

The recommendation here is to keep this project as transparent as possible to upper management:

1. Document CPU savings incurred by the changes. Or, more importantly, any dollar savings associated with those CPU savings. Keep upper management fully informed on both the effort and the benefits associated with this migration.
2. Keep track and publish some level of *percentage complete,* at both the application level and overall. This will become particularly important since at some point the strategy of *migrating as programs come up for changes* will experience diminishing returns – the same programs tend to come up for changes over and over. This will help your organization define the meaning of "done" in regards to Phase 2 of the migration.
3. Record any issues as they come up. Again the recommendation is to be as transparent as possible as this project is tracked upwards within your organization.

Challenges in Phase 2

- Expect “devil is in the details” type problems at this point.
- Exception criteria?
- When to move to Phase 3? At what percent complete is Phase 2 considered “done”?

Figure 4 Phase 2 Challenges

Phase 3

Aggressive Changes

Once you are comfortable with the processes and success of Phase 2 changes, the recommendation is to now begin implementing changes that will realize the highest return – regardless if these programs are being changed for other reasons or not. For instance you may review your R4HA charges to identify likely tuning candidates, identify high volume CICS transactions, IMS transactions or DB2 stored procedures or very long running batch jobs. Again you should decide how aggressive to pursue performance improvements (for these high impact programs – do you use more aggressive compile options?).

In this phase you may also decide to do complete regression testing to ensure errors are accidentally introduced or revealed during the migration.

Remember saving MIPS is beneficial but it may not result in dollar savings within your organization until you can reduce your rolling four hour averages.

Phase 4

Phase 4 involved retiring your previous COBOL run time. At this point you should be very confident that any program in your portfolio can be converted to COBOL V5.

Phase 4 Considerations

- Create a clear definition of when Phase 2 is done (at what code percent complete).
- Create a clear definition of when Phase 3 is done (what is the cutoff for programs to be considered in the CPU intensive list addressed in phase 3?).
- Phase 4 should include a declaration by all involved parties (application, systems, SCM administrator) that they are ready to retire the older COBOL
- Thoughts should be given to steps moving forward once the project is done.
    - Identifying and migrating very old (VS COBOL II, OS/VS COBOL) programs?
    - Identifying and migrating remaining COBOL programs?
- A project analysis should be done (post-project) to aggregate efforts and benefits with the project.

Figure 5 Phase 4 Considerations

Summary

This write-up recommends a four phase approach to a COBOL V5 migration strategy.

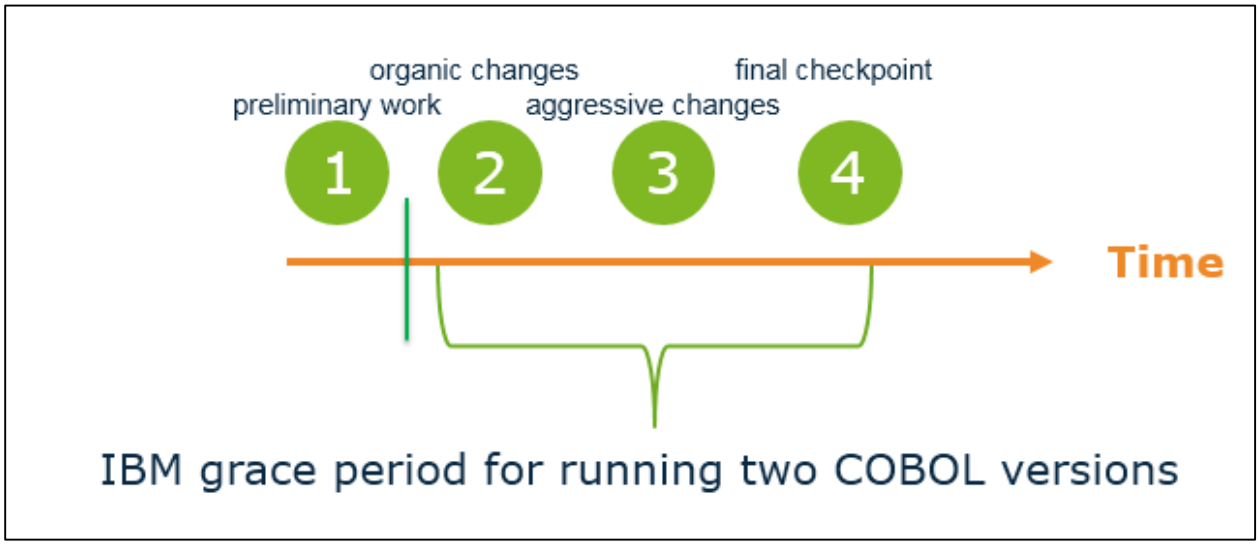

Figure 6 Four Phase Migration Project

This write-up also recommends a cooperative effort across three specific groups: application, systems and SCM admin.

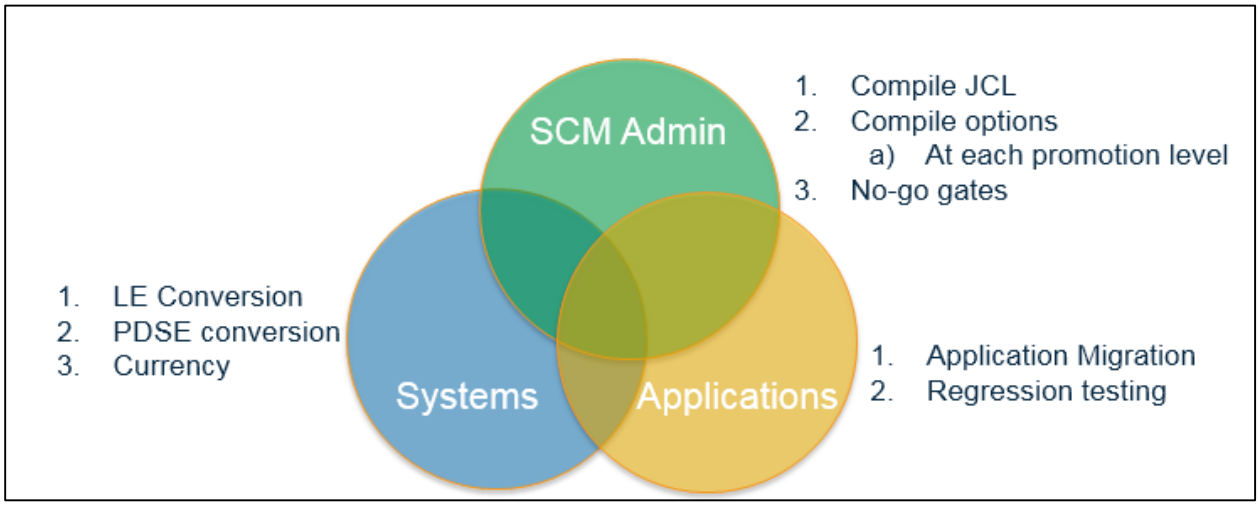

Figure 7 Coordinated effort across three groups

A carefully planned migration project should ensure that expectations are level-set prior to any of the work; that progress is transparent to upper management; and that the migration is accomplished as smoothly as possible and with the greatest possible benefit.