The IT Knowledge Gap

How to reduce the mainframe learning curve as the new generation comes on board

One of the most persistent issues for IT is the ever-growing knowledge gap among IT practitioners—those who configure, maintain and run z/OS*, and are the core of any reasonably large enterprise. I say "a knowledge gap" rather than the oft-mentioned "skills gap" because young (and not-so young) short-tenure system programmers are often quick to pick up the skills, but the insight into how the systems hold together—particularly the interactions with their own customization—is much harder to come by.

The problem is aggravated by what might be called the age gap among z/OS system programmers, which was caused by those premature reports of the death of the mainframe in the early 1990s. Most practitioners are either over 50 or under 30. Vast amounts of knowledge are disappearing as the older generation leaves the industry to be replaced by bright new people who just lack the institutional knowledge of those they are replacing.

The situation is worsened by the tried and true attitude of the mainframe community that, "If it ain't broke, don't fix it." This leaves in place elements of the environment that were introduced decades ago and are still in place today. Knowledge of these elements is constantly evaporating, and in many cases, has been completely lost. It's very hard, and potentially dangerous, to change or enhance something that isn't fully understood.

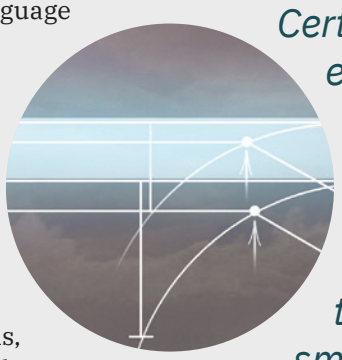
A good example of this would be the number of z/OS installations that have remained with JES3 despite it being under threat for decades. JES3 worked for them so they stayed with it—but now the situation requires action.

Tim Humphreys, founder and president of Trident Services Inc., applies his 45-plus years of IT experience in providing software and services to augment in-house system programming capabilities.

Why It Matters

It's unfortunate that z/OS and related products still require that

certain customizations be done by writing Assembler language code to direct the systems to do things the way the installation wants. Many tools can assist with the understanding and incremental enhancement (now called modernization) of old core applications, but tools to help with the few bits of system-level Assembler code that every installation has are scarce, if not non-existent. The amount of code is not the problem, but rather the fact that it runs in an authorized state such that an inadvertent error can crash the whole system or a bit of maliciousness can undermine the entire enterprise.



Certainly, we must make every effort to properly educate our "next gen" co-workers, but we also must make it less of a herculean task for them to get up to speed on over a half century of technological accretion. There's expertise out there that can be acquired that will keep an installation running smoothly despite the gaps.

A plethora of other customization and tailoring opportunities are also provided that aren't so inconvenient as actually requiring Assembler language code. Many of them go back to the days when computer time was extremely expensive and installations sought to do any tweaking they could to get the most out of their MIPS. Many of these options are unnecessary and some may even be somewhat dangerous.

What Can Be Done?

Obviously, intensive education can provide some relief, but it's expensive and time consuming. It's hard to cram decades of knowledge into a reasonable timespan—and it may not be worth the cost when other possible mitigations exist.

Another approach is to outsource parts of the system programming efforts to service organizations that already have deep expertise in specific areas. This can be done either on an ongoing basis or as a one-time modernize and document effort. The local staff can move forward from there. Product migrations like moving to a new release, converting from one security product to another or moving from JES2 to JES3 could easily be viewed as not being worth the education time, were they to be done by the local staff. These areas of system performance and capacity planning require very high levels of knowledge and experience and might benefit from some outside expert consulting.

A third approach is to use ISV tools to simplify the system programming tasks. One example would be products that raise the activities to a higher level of abstraction. This allows the system programmers' knowledge to be geared toward implementing installation policy without having to understand, for example, JES2 internals.

Looking again at system performance and capacity planning, if it's being done by local staff, their efficacy can be greatly increased with the help of tools that provide easily consumable data about what's going on performance-wise in the system. This can be provided by products that run in house or by remotely provided services, and might even save money by forestalling upgrades.

Cost, Cost, Cost

An often paramount consideration for mainframe installations is cost. As outlined, some savings can be had in an environment of decreasing knowledge with some help from the outside. But many other ways exist to reduce cost with some outside help.

The way software is charged on z/OS is quite complex, to say the least. Installation staff attempt all kinds of things to lower that magic rolling four-hour average. Although often effective, some of these approaches risk either missing performance goals or violating software license agreements. As the complexity of the systems (and the pricing rules) increases, it might be best to get some expert assistance from software products that attack the problem with the diligence and accuracy of computer programs.

Products are available that reduce cost "the honest way" by actually reducing CPU usage, often by eliminating unnecessary system overhead. Others take what might be called a "girdle approach"—that is, they push work from one place to another without actually reducing the amount of work accomplished. Due to the specific software licensing rules, this type of approach can be highly effective in reducing costs. All of these approaches are complex and require deep understanding to execute without causing responsiveness problems, or worse.

Filling the Gap

Certainly, we must make every effort to properly educate our "next gen" co-workers, but we also must make it less of a herculean task for them to get up to speed on over a half century of technological accretion. There's expertise out there that can be acquired that will keep an installation running smoothly despite the gaps.