NEWS FLASH: MAINFRAME BATCH IS CRITICAL AND MUST BE AUTOMATED!

I recently had an epiphany, a lightning strike of realization of something I surely knew long ago but never really registered: **Mainframe batch is a manually operated environment!**

For those of you who have worked with the mainframe for decades, you may join me in feeling that we have been too close to the forest to see the trees. Manually operated batch was just how it was, how it worked; you didn't question it.

Okay, that's how it was. Why is it still like this? It doesn't have to be, and it certainly shouldn't be as mainframe teams are required to accelerate not just development but also bring agility to operations. Automated batch processing is a critical part of that.

Every mainframe shop runs at least a modicum of batch. Many shops run fantastic amounts daily, in the hundreds of thousands of jobs. Almost every shop has a job scheduler to organize what runs when—a herculean feat in and of itself. But then what?

Batch 101

IBM provides two spooling systems: JES2 and JES3. Both evolved from the early mainframe days to provide the first virtualization of hardware resources, which enabled a computer to multitask, working on multiple business processes concurrently. Each accepts jobs as input to be executed and queues them up for processing by job class.

Then special address spaces called initiators, request work from JES; requests are for the next job in the queue, with the same class as assigned to that initiator. JES then hands them the job, which the initiator digests and proceeds to execute. Print output is handed back to JES to similarly queue for transmission elsewhere, storage on tape or disk, or physically or virtually printed.

Not So 101 After All

This seems simple enough, but not when tens or hundreds of thousands of jobs a day must be processed. How many initiators are needed and where? What class(es) should you assign to each initiator? What if a job abends making a series of jobs late? How do you stay on schedule?

The phone rings and a new priority emerges. Someone enters commands to adjust initiators for that. Elsewhere, a programmer made a mistake, and a whole job stream must be rerun before subsequent jobs can execute.

Someone must juggle the queues to accommodate. Someone must start initiators, or stop initiators, or change their job classes. Another abend requiring a call to get somebody out of bed to resolve. Someone must hold this job; someone must release that job; another special request comes in!

And this madness is all on a *good* night.

What's Your Mainframe Batch Strategy?

Every shop has a batch strategy. It revolves around all manner of soft constructs like workload types (production, test, QA, datacenter management, slow-boat-to-China, you name it); job classes; component naming standards: job names, job step names, program names, dataset names, DD names, procedure names; security rules; Sysplex architecture, LPARs, databases, queuing engines and many other things in your shops, not mentioned.

Systems programmers and lead operators with years of experience, who probably grew up with your company as your mainframe environment grew, devised your batch strategy. And more importantly, they continued to evolve it as your business needs changed.

Standards were meticulously developed and published; capacity plans for batch resources were conducted; initiator and job class structures were designed and constructed in JES; exits were coded to enforce standards and other activities were managed, all requiring hard-won technical knowledge, skills and experience.

It's not rocket science—it's harder than that.

IBM gave their customers a blank slate of fundamental structure and controls and essentially said, "Good luck with your problem. Do the best you can." And requests for guidance were always answered with, "It depends."

So, sysprogs and operators rose to the challenge. They did the best they could with their knowledge, skills, talents and time available, and they chiseled out of solid granite a strategy and fundamental controls. IBM just put their customer into the systems software business.

What Will Happen to Your Strategy?

Who is going to do this when the people who designed and built it, and work every day to operate it,

retire? It takes all their accumulated knowledge and experience to do the job now.

How will someone with less technical knowledge of JES and its exits, assume that systems programming challenge? Do they have assembler skills? Do they know the intricacies of JES execution? Control blocks – in memory data structures the initiators use to run the job? Expertise in WLM?

Maybe you can call IBM. Wait! They have the same gray-hair problems, too.

How will you replace the lead operators who know which job must be complete by 10:25 p.m. or a cascade of missed deadlines will ensue? Who observes when development modifies the accounting system's batch stream, and the critical path has changed? It's only one of thousands of job streams.

Who has been tested by fire, time and time again, and rises to the challenge? Who is ready for that? Are new, hopefully youthful people understudying for *years* to assume the mantle soon to be handed to them?

Bleak or Bright: It's Your Mainframe Batch Future

You may think I paint a deliberately bleak picture with these incessant questions. But when organizations rely upon the accumulated experience of their highly specialized people, keeping their headcount as low as possible, there will be dramatic gaps when those people depart. With so few people, they have to use their years of experience to cut corners where they can and laser focus on the crux of the problems.

There is no practical way that less experienced people will take over and succeed without hiccups, if not disasters. Experience is what you get when you didn't get what you wanted.

But allow me to lighten the tone now: It's not hopeless. There is an option: Automated batch processing.

IBM may not have spent 30-plus years and written 2 million lines of code to enable automated batch processing, but a little-known company in Canada, MVS Solutions, did. They saw the early problems and they grew their solution as IBM's batch failed to scale, while customers' workloads grew and grew.

This solution is ThruPut Manager, a rules-based and policy-driven control system providing for the complete automation of batch processing in a JES2 environment. Today, BMC AMI DevX is continuously innovating this inimitable tool that enables automated batch processing, having acquired MVS Solutions in 2017.

I've worked with ThruPut Manager for 20 years, and I can say it will solve every batch problem in your shop—even the ones you don't yet know you have.

IBM has announced stabilization of JES3
ThruPut Manger does not operate in a JES3 environment, but it can process JES3 JCL on JES2.