FALL 2020

Control-M

From BMC

by Stephen Swoyer

BG

Building a Foundation

Companies that are successful with data science tend to share three key traits: they're highly agile, they're customer-focused, and they're data-driven. This is one of those cases in which correlation really does rise to the level of causation. If you're data-driven and customer-focused, you're almost by definition agile. As a function of each of these traits, you'll likely have success with data science, too.

Q.E.D., as the philosophers like to say.

The problem is becoming any one of these things is easier said – or wished for – than done. You can't make a slow and plodding organization agile just by issuing an edict, purchasing a technology, or implementing an architecture. You can't snap your fingers (or buy a technology) and make a company that lacks insight into its customers and their behaviors customer-focused. And you can't take a decision-making culture that isn't firmly grounded in data and analytics and – voila! – recast it as data-driven. It doesn't work that way. You've first got to prepare a foundation for these things.

Thankfully, we now have a good idea what this foundation consists of – that is, of its enabling pillars, so to speak. The agility, customer-centricity and data-driven emphasis that are the keys to success with data science are the products of a handful of revolutionary innovations. These are:

- Orchestration
- Enterprise DevOps
- Data-driven Business
- Observability and Adaptability
- The Gestalt

Orchestration. An ability to automate and orchestrate at multiple layers: first, that of the business services that constitute core processes; second, that of the workflows that feed the business applications which undergird these processes. And third, emphasis on eliminating gaps that obtrude in or between processes, especially those that require token human oversight.

Enterprise DevOps. DevOps fuses the chasm between software development and operations, two traditionally discrete processes that frequently clashed with one another. In the last few years, similar fusions – DataOps, MLOps, and AIOps, to name a few – have emerged. DevOps remains a site of transformation, as the rise of site reliability engineering (SRE) demonstrates.

Data-driven business. Before you can make decisions on the basis of data – and before you can automate certain kinds of decisions – you've got to create an infrastructure that orchestrates the acquisition, preparation, and delivery of data. Not only is this essential for traditional business applications and services, but it helps DevOps, DataOps, MLOps, etc. teams rapidly train and deploy AI-infused applications, new curated data feeds, and other critical assets.

Observability and Adaptability. Observable applications, services, and systems permit insight into the availability, reliability, security and integrity of the specific business functions they provide. Observability at this level constitutes a radical departure from reactive practices that emphasize monitoring to detect anomalies in the applications, services, etc. that comprise business functions.

Observability is a prerequisite for adaptability, i.e., the ability of a business function to act on – i.e., to adjust or correct: to "fix" – itself in real time. Along with automation and orchestration, observability and adaptability are critically dependent on data and analytics. Under the covers, an "observable" service or application, is enabled by data flows that feed ML-powered AI services which enable rapid diagnosis and automated correction of potential issues. Adaptability of this kind is a prerequisite for managing not just the health but the security of applications and services in modern application and data architectures.

The Gestalt. The gestalt of all of these things constitutes the foundation for the agility, customer-centricity, and data-driven business operations that translate into success with data science. Each of these pillars complements and builds upon the others. You can't have enterprise DevOps – still less, DataOps, MLOps, or AIOps – without orchestration between and among business applications and services, data pipelines, and the processes in which they're embedded. You can't have data-driven business operations without continuous development and integration practices that create and feed business applications and data pipelines alike. And you can't have observability or, still less, adaptability without all of the above.

Orchestration: beyond automation

“Orchestration” is a deceptively simple term that elides a staggering amount of behind-the-scenes complexity. The thing is, orchestration itself is not a new concept; in fact, it’s something that most of us have long taken for granted. Even though we treat invoicing, billing, and payroll as distinct business processes, they’re actually interpenetrated with one another in the sense that each consumes data from and communicates with API endpoints exposed by the same ERP, MRP, MRP, HR, etc. “systems.” These systems interoperate with and exchange data with one another, sending requests, triggering and receiving output from jobs, and receiving as well as responding to requests from other systems

In a sense, the interaction between the billing and payroll modules of an ERP system is a hugely simplified instance of orchestration. Something to keep in mind is what we mean by the word “system” has changed radically since the appearance of the first all-in-one ERP packages in the late-1980s. It used to be when we called something a “system,” we meant some combination of hardware and software coinciding in physical space: the big black box running DB2, CICS, and SAP in the raised-floor mainframe room; this 4U x86 server running Oracle on that physical rack in the on-campus server room. Today, these “systems” span different contexts, from the on-premises enterprise, to the managed cloud, to (in some cases) the public cloud, with other types of deployments (on-premises private cloud, virtual private cloud) sandwiched in between. (This is also true of the venerable mainframe, which is a huge cloud business for IBM.) The upshot is the “systems” of today span different (regional or geographical) spaces and different (regional or geographical) time zones, too.

Conceptually, distributed ERP, CRM, HR, and other “systems” are, in effect, volatilized. They have no “place” in physical space. In modern architectures, ERP, CRM, and other core business systems span contexts: some resources (finance) still live primarily in the on-premises enterprise; others (CRM) live primarily in the SaaS or PaaS cloud. Orchestrating a workflow that schedules operations in or fetches data from finance and CRM “systems” involves operations that span multiple contexts. Orchestrating workflows and data pipelines on a single platform (for example, a mainframe) or in a single context (the on-premises enterprise) is a relatively straightforward proposition. Physical, on-premises systems have known capacities, known constraints, and predictable performance envelopes. However, this usually is not the case with virtualized resources – especially in cloud contexts, where (owing to the fluctuations of virtualization and remote connectivity) few service providers offer performance and availability SLAs. On balance, then, performance in cloud contexts is predictably unpredictable.

Orchestration can be complicated...

Ironically, most software resources provide built-in scheduling or workflow automation facilities. However, these facilities usually aren't designed for robust interoperability with third-party software. Vendors are incented to promote certain kinds of interoperability (e.g., data ingest into, or data/ system migration to their own software) and to deprecate others (data export/egest out of, data/system migration to another vendor's software) – which complicates the task of orchestrating workflows between business software in heterogeneous environments. But more importantly, the automation facilities built into most software lack a synthetic view of workflows across all contexts. The system only knows (has insight into) what's happening in its own context.

This is true, as well, of emergent paradigms such as microservices architecture and function-as-a-service (FaaS, also known as "serverless") computing. Both paradigms are predicated on the concept of service and/or workflow orchestration – i.e., orchestrating workflows of containers or code that provide fine-grained "units" of application functionality. Kubernetes is the most popular software for orchestrating services in microservices architecture. (Kubernetes also sees use – via, for example, Knative or Kubeless – in FaaS/serverless computing.) But neither Kubernetes nor its ecosystem of enabling packages (Helm, Prometheus, etc.) achieves anything like a unified view of service or workflow orchestration across all external applications, services, systems, platforms, or contexts.

For example, you can use Kubernetes to orchestrate a collection of microservices that fetch data from a CICS transaction gateway; kick off an ETL job via DB2 running on the same mainframe system; integrate the output of both jobs into a data set; transform this data to JSON: and fetch said JSON to support an AI fraud-detection service. The problem is that Kubernetes lacks insight into what exactly is happening in the mainframe context; it is solely responsible for the services it orchestrates – not for the jobs these services trigger on third-party platforms, and still less for what's happening with the underlying software or hardware resources on these platforms. If a dependent job should fail, Kubernetes cannot determine why this happened, nor can it take steps to compensate for any errors. It can only kick off an alert and reschedule jobs. The resulting stall effectively breaks the data pipeline.

...More Complicated Than You Think

IStalled data pipelines, in particular, are a ubiquitous problem. This is a function of the essential distributedness of data, which is a feature – not a bug! – of the way data "happens:" i.e., the typical circumstances in which data is generated, exchanged, and consumed by humans and machines.

Today, the data people and software require is created and distributed across disparate contexts: some in on-premises ERP systems and databases, some in SaaS or PaaS cloud services, some in the IaaS cloud, some on the web, some walled-off by VPN gateways and/or proprietary mechanisms.

Thus the rub: if data is valuable in and of itself, it's even more valuable when it's combined with data that complements or extends it in some way. This is the essence of analytics: i.e., the idea of combining data from different sources to represent a multifaceted view of something about the business world.

The upshot is that creating data pipelines involves orchestrating operations between and among contextually and geographically distributed sources of data. It involves different kinds of access (ODBC/JDBC; SMB/SSH; RESTful APIs; vendor-specific APIs) as well as translation between different exchange or serialization formats. Pipelines always entail logic, too. Because pipeline dependencies must be sorted and anticipated in advance, this sometimes results in complex rules and decision trees. But creating distributed data pipelines is arguably even more complicated: logic and rules for different kinds of failure (service unavailability; abnormal latency; incomplete or corrupted output; etc.) must be created and tested, too. Because of the greater likelihood of failure, logic for recovering data from completed operations (instead of restarting the pipeline) is more robust.

Last, orchestration is almost always a team effort, which places a premium on collaboration. Teams require a single context in which to collaborate, with each team contributing steps and logic to its portion of a workflow – as well as commenting (and contributing to) the work of other teams. Developers and architects likewise require some means of merging, reconciling, and – most importantly – testing and validating all of this collaborative work. It's fair to say that the essential distributedness of data poses the single biggest challenge to orchestrating workflows – not only for data pipelines but for business applications, too. After all, the ability to orchestrate interactions between and among business applications likewise presupposes the ability to orchestrate interactions between and among data pipelines.

To sum up, the distributed nature of data pipeline workflows creates the equivalent of gaps between systems and contexts. To orchestrate workflows, IT specialists and data engineers often opt to bridge these gaps using the same tools they've always used: shell scripts and facilities such as the ubiquitous Cron. Complimentary to the growth of microservices architecture and similar paradigms, they're also exploiting new technologies (such as container virtualization and microservices orchestration) to bridge gaps. But these approaches lack visibility into the orchestration of workflows across heterogeneous systems and contexts. They require a variety of software tools to monitor and control workflows and operations.

What's needed is a comprehensive view of applications, services, and systems – as well as of their underlying resources. One example of this is Control-M, an orchestration technology from BMC Software.

CONTROL-M FROM BMC

AN OVERVIEW

Control-M provides a layer of abstraction with respect to the creation, management, and orchestration of the service workflows that power essential business processes. In fact, it provides two distinct layers of abstraction: first, with respect to the topology, status, and health of essential business services, such as the order creation and order approval services that (among others) power the business procurement process. Second, with respect to the workflows that undergird these and other services. This is important because different roles or personae require different levels of abstraction. A business process engineer or business manager is less interested in the nuts and bolts of (to refer to the previous example) procurement than in the performance, quality, and availability of the business services integral to this process. She needs abstraction at the layer of (and control over) the process itself. This is her area of expertise.

There are conceivably cases in which software and data architects, along with DevOps and SRE engineers, will benefit from abstraction at this level, too. For the most part, however, the DevOps or SRE engineer requires a different, more granular, kind of abstraction. If they are designing a new workflow or troubleshooting an existing one, they need a means to peer beneath these business services to illumine the core applications, services, systems, and platforms that constitute them. For example, if a data pipeline stalls because a stored procedure in an Oracle database is not triggered for some reason, they need to be able to diagnose and fix the problem. In some cases, they will also need to peer into the host environments in which these resources "live." Not just into z/OS, AIX, HP-UX, Windows, or any of several flavors of Linux, but into PaaS services and core cloud infrastructure services – Amazon AWS, Google GCP, Microsoft Azure, IBM Managed Extended Cloud, and similar. In the cloud context, especially, even the smallest change – the deprecation of an API; an update to a library or infrastructure service that introduces new dependency requirements – can break a workflow.

The Challenge of DevOps

Control-M is already used by business process managers, analysts, directors, and engineers to stay informed on how business services are performing, much like they would track the status of a flight from a mobile app.

In the same way, Control-M is also used by operations teams to manage and choreograph the delivery of business services. But thanks to the fusion of software development and operations in practices such as DevOps and SRE, Control-M's operational capabilities can now be embedded into a DevOps toolchain. It's important to note that Control-M is not orchestrating the DevOps toolchain but instead it's workflows can be built and deployed using a CI/CD model.

Developers can use Control-M's built-in visual editor to (for example) drag, drop, and connect the discrete steps or operations that comprise a workflow. But BMC has evolved Control-M to accommodate the practices, methods, conventions, and expectations of developers who live and work in the DevOps and SRE models, too. For example, developers can use a Jobs-as-Code approach now invoke Control-M's RESTful automation APIs in order to schedule and manage jobs across software that lives in different contexts. In addition, BMC exposes RESTful automation APIs that provide access to Control-M's built-in verification, testing, and execution functions. This makes it possible for DevOps teams to use it to automate the unit testing and validation tasks that are core components of the build process.

This is all thanks to Control-M's new Jobs-as-Code facility, which was conceived with DevOps and SRE in mind. Developers can now create and orchestrate Control-M workflows in JSON code, as well as persist that code to a source control-management (SCM) system, such as Git. In this way, Jobs-as-Code supports the continuous integration and continuous delivery practices that are the trademarks of DevOps. With Jobs-as-Code, DevOps engineers can work in their preferred environments - using familiar editors (e.g., vi[m], Emacs); familiar SCM or revision control tools (Git); familiar languages (JSON) and language runtimes (node.js). BMC also provides a virtualized test-dev sandbox (Control-M Workbench) that developers can use to experiment with Jobs-as-Code if their organizations do not have an existing Control-M implementation. Jobs-as-Code also extends its scheduling and orchestration capabilities to AWS Lambda, the first and most prominent FaaS platform.

The Priority of comprehensive visibility and control

For DevOps and SRE engineers, then, Control-M provides a comprehensive view into – and control over – workflow scheduling and orchestration across distributed applications, services, systems, or platforms, irrespective of context. It enables the equivalent of a single point of control over workflow orchestration in highly distributed software or data architectures. What does this look like? And why is it important?

An earlier example described a workflow that (1) grabs CICS data from mainframe VSAM storage; (2) integrates it with data extracted from DB2 running on the same mainframe; (3) persists the output of this integration job to a JSON file; and (4) moves the data to a public cloud to use a serverless PaaS tool to process the data and then move it to a cloud-native data warehouse for analytics. Imagine that this same workflow spans not just the on-premises enterprise, but the managed cloud too. This radically increases the complexity factor, requiring that DevOps engineers design data pipelines that orchestrate operations between distributed platforms and application contexts. Assume, too, that the CICS data is sensitive: it cannot move outside of the on-premises context and must be masked before it can be consumed by the fraud-detection service.

In this example, the workflow fetches data from a mainframe DB2 instance hosted in IBM's Managed Extended Cloud and uses an on-premises mainframe to integrate it with masked CICS data. The performance of the on-premises mainframe is tightly controlled and highly predictable; that of the cloud mainframe service is also highly predictable – but nonetheless subject to the fluctuations of VPN access. In practice, the data pipeline could stall because of a problem with the VPN or with its enabling Internet connection.
It could stall because of a problem on the public cloud.. It could stall because of a problem with DB2 running in the mainframe cloud instance. And it could stall because of any number of unpredictable problems in the on-premises context, too. In order to diagnose and fix a pipeline stall – and, moreover, to build adaptability into the data pipeline itself – engineers need insight into a plethora of applications, services, and platforms. These could include:

1) *Kubernetes running in both (virtualized) x64 Linux and zLinux 2) CICS running under z/OS in the on-premises mainframe context 3) DB2 running under z/OS in both the managed cloud and on-premises mainframe contexts 4) The virtualized microservices in public cloud that constitute the AI fraud detection service.*

Engineers could build services that monitor software resources across each of these distributed platforms and contexts. They could build services that detect anomalies, and which relate these anomalies to the failure of specific operations or dependencies in a pipeline. And they could build services that identify the root cause of service degradation or failure, and which determine if it is practicable to recover (as distinct to restarting) a stalled pipeline. Services that, finally, trigger one or more actions to fix these problems. This is possible, even practicable. Engineers and architects do it every day. But should they?

Conclusion: Technology such as Control-M addresses two critical gaps

This results in incredibly complex workflows with an excess of moving parts. A large portion of this complexity stems from the problem of monitoring and automating workflows across distributed application, service, system, or platform contexts. Absent a comprehensive means to view and control workflows across software in all contexts, engineers spend a disproportionate amount of time building services that reduplicate or exploit scheduling capabilities that are already exposed by x64 Linux, zLinux, z/OS, CICS, DB2, and Kubernetes, which runs in both the x64 Linux and zLinux contexts.

All of this is unnecessary work. Software such as Control-M is available for all common operating environments – including legacy systems such as OpenVMS -- all common databases, application servers, and container orchestration managers as well as modern PaaS offerings on public and private clouds. It can be deployed on these resources in either the private or the IaaS cloud. This breadth of platform coverage gives business process engineers, ops teams, DevOps engineers, and others a means to knit together the workflows and data pipelines that power essential business applications and services into a single comprehensive view.

Software such as Control-M addresses two critical gaps in the status quo. The first is that of conventional, ops-oriented technologies that abstract at the level of business services – but which afford scant visibility into (and little to no control over) the workflows and data pipelines that constitute these services. The second is the symmetrical opposite of the first: i.e., ops and, especially, DevOps-oriented solutions that abstract at the nuts-and-bolts layer of applications, services, systems, etc. but which provide neither a complete view of (nor control over) business processes and their services.

Control-M's comprehensive view is suitable for a variety of user personae: its graphical features permit ops teams can quickly identify and diagnose problems with workflows – or, if necessary, collaborate with business process engineers to reconfigure impaired or unavailable services. Similarly, DevOps engineers, developers, and personae can build, test, validate, and orchestrate service workflows using Control-M's built-in tools or – via its FaaS-like Jobs-as-Code facility – using their preferred toolchains.

The upshot is that software such as Control-M not only permits business managers and ops teams to manage, optimize, and orchestrate services (and their constitutive workflows) at a high level, but – at an altogether different level of abstraction – helps accelerate the building, testing, validating, and delivery of workflows in consonance with continuous integration and delivery practices. It knits together two worlds, and two wildly divergent experiences, that are otherwise disjunct.

ABOUT THE BLOOR GROUP

The Bloor Group is an independent research firm that produces objective, high-quality analysis of enterprise technology products, services and markets via new media outlets and traditional research methods.

As a hybrid New Media/Analyst firm, The Bloor Group seeks to educate business and IT professionals on the array of technologies and methods available for managing information, and provide innovative vendors with the resources to promote enterprise software and services.

BMC helps customers run and reinvent their businesses with open, scalable, and modular solutions to complex IT problems. With unmatched experience in IT management, we support 92 of the Forbes Global 100 and have earned recognition as an ITSM Gartner Magic Quadrant Leader for five years running. Our solutions offer speed, agility, and efficiency to tackle business challenges in service management, automation, operations, and the mainframe.

ABOUT BMC

BMC helps customers run and reinvent their businesses with open, scalable, and modular solutions to complex IT problems. With unmatched experience in IT management, we support 92 of the Forbes Global 100 and have earned recognition as an ITSM Gartner Magic Quadrant Leader for five years running.

BMC's solutions offer speed, agility, and efficiency to tackle business challenges in service management, automation, operations, and the mainframe.