bmc

Thought Leadership White Paper

Workload Change: The Big Part of Your Business That DevOps Forgot

Accelerate your software and application development process with workload automation

By **Joe Goldberg**
Innovation Evangelist, BMC

Table of Contents

1 EXECUTIVE SUMMARY

2 WORKLOAD AUTOMATION IN THE ENTERPRISE

DEVOPS

Dev

Ops

So What?

3 IS THIS UNIQUE?

WHAT IS NEEDED?

The Attitude

The Functionality

4 THE BRAVE NEW WORLD

5 CONCLUSION

Executive Summary

It's almost universally accepted that applications built today must be delivered and updated faster, and most likely retired faster. The current ideal state of application development is continuous delivery, where change and improvement is constant. The problem for enterprise IT is that change disrupts stability. However, authority and budgets reside with the business, and what the business wants, the business usually gets.

Constant change has led to the rise of DevOps and a focus on communication among the various groups and disciplines that make up the entire software development lifecycle. **One mystery, however, is the almost complete absence of workload automation or batch processing in the DevOps discussion**. But what's really the value of batch processing in this world of real-time updates? Payroll, inventory, supply chain management, logistics, bank deposits, customer sentiment analysis, web store recommendation engine infrastructure, and big data actually are all predominantly, if not completely, batch.

Batch is not going away. Batch makes up the majority of business processing performed by computers, and that share will get bigger. This is because interactive processing is getting more complex, requiring more computing power, and leaving fewer resources for doing the stuff that doesn't need to be done in real time. For example, if you have ever booked an airline reservation in real-time or bought something at a web store, you may have received an email later (up to 24 hours later for airlines) with the actual confirmation. That's because there's a lot of processing that must be completed before the actual confirmation can be made—and you wouldn't want to wait in front of your computer for that confirmation. Instead, it is completed in batch, and you're notified later.

The goal of this paper is to discuss why workload automation, commonly known as batch or job scheduling, is largely missing from the DevOps discussion and to explore some possible ways to remedy that situation.

WORKLOAD AUTOMATION IN THE ENTERPRISE

When examining the typical collection of technologies that make up an application in the production environment, workload automation is unique. Let's say a complete application stack consists of an OS such as Linux®, an application server such as Apache, and a relational database—Oracle® MySQL® might preserve the LAMP stack concept, but in traditional IT, it would most likely be Oracle® RDBMS. Other components may comprise messaging infrastructure and perhaps a "plain" HTTP web server.

Developers develop in their language of choice and start testing. Their test environments normally include the entire application stack. For enterprise apps, testers are running only portions of the entire application and thus using only a portion of the application stack. However, once unit testing is complete and system testing begins, the entire stack eventually is deployed and utilized. This all happens as part of the software development life cycle (SDLC), and many developers get comfortable with that technology stack.

System, user, and acceptance testing eventually is all complete and it is time to promote to production. Once the application arrives in this new environment, some technology may be applied that has not touched this application previously. It's likely that there is a monitoring solution that now watches for storage, processor, and memory consumption. Database administrators may be watching the complexity and efficiency of SQL queries, and network admins may be examining the load this new application places on network bandwidth.

Generally, these are operational tools that watch the application in the context of the overall IT environment and monitor the health of the technical components. **If any or all of these tools were to disappear, there would certainly be severe implications on quality of service, maintainability, and capacity management, but the application would continue to run**.

That would be the case, unless it was a batch application or had a batch component and one of the solutions that disappeared was workload automation. Ask any IT operations practitioner what happens when their "batch scheduler" stops and the universal answer is "our business stops!" **The business of IT is clearly linked to workload automation. Workload automation is a critical part of the application, yet it is usually perceived as exclusively an operations tool**.

Developers are very familiar with the application technology stack, but they are usually completely unaware of workload automation. Even though the success of an application will eventually depend on how the workflows are built and how efficiently they run, the connection between application code and workload automation is not made until a relatively late point, often at the very end of the SDLC.

DEVOPS

DevOps is a software development method that responds to the interdependence of software development and IT operations with a focus on communication, collaboration, and integration. The end goal is rapid, efficient production of software products and services. It's considered a culture or process upheaval that seeks to address the disconnect and the traditional "throw it over the wall" approach to application deployment.

It is not intended to significantly change roles or disrupt processes, but rather to focus on greater communication and transparency. Thus, when part of the application deployment includes a web application server configuration or the need to build a database instance or update schema, DevOps seeks to automate, streamline, abstract, and communicate those activities. However, if a step or activity is absent from the current manual process, it is unlikely to be introduced in the automated version of that process especially if it requires skill or knowledge absent in one domain (Dev) or the other (Ops).

Dev

In this context, it doesn't really matter whether application development is done using waterfall or agile approaches. Developers write code and do some level of testing. At that stage, the workload automation challenges that will be faced later already make an appearance.

Let's say we are developing a web application that processes various transactions, and after 'n" number of those transactions, spawns or triggers batch processing against the accumulated data. There's code that runs on an application server, transfers the collected data to a location for processing, publishes a message to a message queue to kick off the batch processing, and performs the batch processing itself. All the code generates logs and for some or all of these steps, the log files must be captured for subsequent debugging and analysis. Additionally, the success of one step must be verified before the next step is taken. The first time, or maybe even the first couple of times, the sequence is managed by a developer manually invoking one step, observing the result, and then going on to the next step.

However, this quickly becomes not only boring but inefficient, and scripting is usually applied to address the problem. This decision is what leads to the absence of workload automation in the application developer domain. Scripting comes into play, and as the application proceeds from testing to production, scripts may proliferate and become more complex, leading to problems when it comes time to "operationalize" in the production environment.

Ops

On the operations side, workload automation is viewed as an IT operations tool, so frequently that developers are not given much support or access. It is common to hear of large organizations that have no "test" environments for workload automation. Imagine an application deployment in which a configuration change to some system component is required but has never been tested anywhere. You may shudder at the thought, but it happens with workload automation all the time.

This view of workload automation is entrenched on both the Dev and Ops sides. This results in developers viewing workload automation as outside their domain and their job responsibilities. Like monitoring or network management, it's something for IT to handle.

So What?

Several problems arise from this approach. The first is the amount of time spent on scripting. If one were to examine the shell scripts that become operational batch services, sixty to one-hundred percent of the code can be directly replaced by native capabilities provided by workload automation solutions.

Another problem is the quality of what gets deployed. No matter how good a programmer is, he or she is unlikely to be able to build functionality that will be as robust, mature, and flexible as what is provided out of the box with solutions that have been maturing in the marketplace for decades.

Yet another problem is decreased operational flexibility. Scripts tend to become multi-step, monolithic "black boxes." When something fails, only an on-call programmer can identify even the most basic problems. Once identified, the only recourse is usually complex manual cleanup and wholesale rerunning of successfully completed work. Changes to scripts become new development projects instead of simple configuration changes.

Finally, there is the impact of manual changes as part of the application deployment. This one problem may be the definitive goal of DevOps: **to eliminate manual changes that lead to failures in promotion, and even worse, failure in operation**.

IS THIS UNIQUE?

There are other technologies where the level of developer skills may be deficient even though the technology is commonly used and may even be considered fundamental. One such common technology is relational databases. Many database administrators find the database schema and SQL structure submitted by developers to be woefully deficient. Basic mistakes abound where randomly queried tables are not indexed or a SQL query returns millions of rows when only a small subset is required. Such errors don't surface in testing where volume is small, but become massive bottlenecks in production with fully populated, large databases.

Although these problems do have significant impact, basic database administrator skills are generally considered mandatory for developers. That major distinction is absent when it comes to workload automation.

WHAT IS NEEDED?

How hard is this problem to fix? It turns out that it is quite hard because it requires a change in attitude as well as functionality and tooling.

The Attitude

The view that workload automation belongs to IT operations is almost universal. Even organizations that make workload automation available to developers do not expect them to produce production-ready workflows. If the job definitions and workload automation elements that are built by developers are not production-ready, there is no reason to attach them as artifacts to the deployment process and expose them in DevOps.

To change this situation, workload automation needs to be integrated with the set of developer tools. In order for that to happen, developers must see a benefit to going out of their way to learn something new. That benefit could include a shorter development effort, a higher quality deliverable, or an attractive skill to add to their résumé—although it may take the industry a while to recognize the value of this new skill.

The Functionality

Today, most workload automation solutions are aimed at IT schedulers and operators. The architectures, user interfaces, and terminologies are all oriented towards expert or power users. Building workflows requires not only deep scheduling domain expertise and knowledge of the specific product used, but also the standards and conventions unique to the organization. The entire process is monolithic and does not even contemplate collaboration. In order to draw developers in, a different approach is required.

First, it must be acknowledged that different levels of engagement will be the norm. Just like some developers are experts with relational databases and others are novices, workload automation must accommodate both power and casual users.

Second, the solution must accommodate organizational standards. This is particularly crucial in the world of outsourcing, offshoring, and extreme staff turnover. It's enough of a challenge to become comfortable with a workload automation that may be broadly used throughout the developer community. Thus, it's optimistic to expect highly mobile developers to know anything about the local standards of the organization, especially since it's common for complex environments to have equally complex standards and naming conventions.

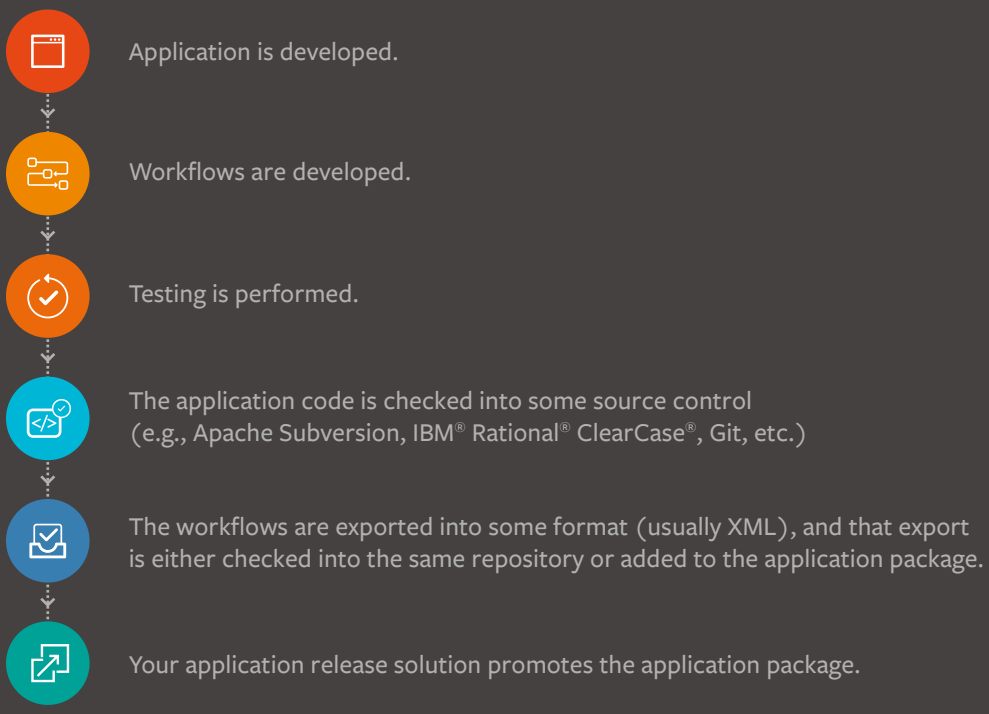
Third, collaboration must be fundamental. No matter how tightly organizations embrace DevOps, there will remain for the foreseeable future two separate groups who need to communicate and collaborate: Dev and Ops.

Finally, the solution must be a high quality "management" application. That means it must be intuitive and easy to use, fully automate the managed function (i.e., building workflows), and contain comprehensive audit and logging facilities to support high levels of governance.

THE BRAVE NEW WORLD

Once developers have embraced workload automation as part of their world and have started generating workflows, the organization is ready to add workload automation to DevOps. The DevOps solution ideally is workload automation-aware and can ingest workflows as a component of the application profile. If DevOps is not there, workload automation can join as another artifact that is part of the deployed application.

Here is a possible series of steps:



This scenario can work well assuming the workflow definitions have already been built in the proper format and are fully formed (or close to it) and ready for subsequent environments.

CONCLUSION

Organizations that have successfully integrated workload automation into their software development lifecycle have seen substantial benefits all the way down the chain. Developers finish their development projects faster because workload automation significantly reduces scripting and facilitates testing. In fact, the more complex the application, the greater the benefits are of applying workload automation to its testing.

The quality of the delivered application is higher because workload automation provides comprehensive operations capabilities like monitoring for success or failure, automatic opening of incidents, ability to send email notifications, and codifying service levels that drive predictive run-time analysis and early warning of potential breaches. Once applications get into production, fewer errors occur because workflow creation has been automated to avoid the errors that result from manual procedures. Automated adherence to site standards eliminates failures resulting from security and similar administrative controls requiring a level of conformance difficult to achieve manually.

Adding workload automation early in the development process ensures that the benefits of DevOps accrue for all applications, including your batch services. Considering that your paycheck or payroll deposit is most likely generated by workload automation, you may find that reassuring.

FOR MORE INFORMATION

To learn more about workload automation solutions from BMC, please visit **bmc.com/it-solutions/control-m**

ABOUT THE AUTHOR

Joe Goldberg is an IT professional with several decades of experience in the design, development, implementation, sales, and marketing of enterprise solutions for Global 2000 organizations. He has been active in helping BMC products leverage new technology to deliver market-leading solutions, with a focus on workload automation, big data, cloud, and DevOps.