White Paper

ChangeMan ZMF
WebSphere

Modern Mainframe Application Development

How to Accelerate Business Change While Minimizing Cost and Risk

MICRO FOCUS

Table of Contents

Today, modernizing mainframe applications means moving more application workloads and functions onto the mainframe.

Executive Summary

The winds of application change are blowing—right into your data center and mainframe environment.

Today, more and more people want instant access to their data on the Web and via their favorite devices. The evolving mainframe application architecture is ready to accommodate them.

Today's mainframe can run Java workloads. It can host Web services, use virtualization to run multiple instances of applications, and function as part of the cloud. Mainframe cycles (million instructions per second or MIPS) continue to get cheaper. Advances such as zIIP and zAAP specialty processors make processing still cheaper by letting developers offload application processes from the general-purpose CPU.

These advancements create a rich palette for application developers of all kinds—both traditional mainframe developers and those working in more contemporary languages.

Businesses have an opportunity to create new applications that combine the stability, reliability, speed, and processing power of the mainframe with the flexibility of Web era interfaces.

Mainframe-quality software configuration and change management (SCCM) can ensure that these new applications are of high quality and work as intended—and that the path to production is a smooth one.

Modern Mainframe Applications

"Modernizing" mainframe applications used to mean moving applications off the mainframe. Today, modernizing mainframe applications means moving more application workloads and functions onto the mainframe.

IBM has done a lot of the heavy lifting by continually evolving the mainframe application architecture to meet business requirements such as e-commerce. The traditional monolithic mainframe application architecture is becoming more flexible and agile and more responsive to business change.

While the modern-day mainframe continues to run legacy workloads—online and batch, COBOL and PL/I—it also runs modern workloads such as e-commerce, Java, and Web services, making it incredibly versatile. The mainframe can finally operate as a giant server in enterprise applications.

All of this is great news because it enables businesses to create new applications that improve service to customers or make the business more efficient than it currently is.

One large Serena (now Micro Focus) customer runs its entire business on applications that combine a browser-based Java front-end with a DB2 back-end on the mainframe.

For example, one large Serena (now Micro Focus) customer runs its entire business on applications that combine a browser-based Java front-end with a DB2 back-end on the mainframe. The mainframe runs the customer's e-commerce and order processing and manages internal application flow. Written in Java, the applications use a continuous deployment model instead of the rigid release model common in many large companies.

Of course, mainframes continue to run legacy applications written back in the 1970s and 1980s, and process business critical batch workloads such as credit card transactions. COBOL is still the predominant programming language. Developers are still writing COBOL programs, building online systems in Customer Information Control System (CICS), and targeting DB2 databases. But today these mainframe databases and subsystems—IMS, CICS, and DB2—can easily handle XML code and Web services.

So the lines continue to blur between mainframe and distributed development environments with the availability of HTTP File Server (HFS) file systems, Linux, and other modern technologies and protocols on the mainframe.

There are big benefits to this blurring. For example, you can now offload mainframe code development to integrated development environments (IDEs) or Windows-based environments. This eliminates the cost of a dedicated Time Sharing Option (TSO) address space for programmers, and gives mainframe developers access to debuggers, plug-ins, and other tools not available in the 3270 programming environment.

Java developers want to develop applications that run on the mainframe but also work from within their favorite development environments—such as Eclipse—and target popular devices such as mobile phones and iPads.

But there are also big risks—most notably a much more complex development environment and complex applications that you're constantly changing to accommodate the business. According to Gartner, documented requests for change (RFCs) can range from 300 per month for a small company to more than 5,000 per week for a Global Fortune 500 enterprise. At show time (production), all code still has to come together—from mainframe to endpoints—and ideally work well, every time, all the time.

Overcoming Process, People, and Technology Challenges

For managers responsible for mainframe application development, the challenge is to accommodate this new world of hybrid applications so that change happens quickly without more risk or higher costs. Specific challenges fall into three areas: process, people, and technology.

- **Process:** Application Lifecycle Management is much more difficult. Applications now involve many more moving parts, many different technologies, people of different skill sets and programming cultures, and critical dependencies that span mainframe and distributed environments. Coordinating application changes across multiple siloed teams is more complex. Deploying code into production is inherently more risky, and preventing "configuration drift" across the enterprise requires a lot of skill and constant vigilance.
- **People:** Invaluable tribal knowledge about the mainframe is walking out the door as mainframe developers retire. Java developers want to work with the mainframe, but they have neither the time nor the inclination to learn the mainframe environment. They also want to work from within their favorite development environments—such as Eclipse—and target popular devices such as mobile phones and iPads. Each camp—mainframe and distributed—has its own way of doing things, and is reluctant to change. Mainframe migrations or business events such as M&As can spawn—sometimes overnight—leaving IT teams in the dark about applications or even where to locate the source code.
- **Technology:** Traditional tools for mainframe software configuration and change management tend to be hard wired into the old mainframe. They aren't easily adaptable to modern mainframe application development—including concurrent development methodologies like Agile or Scrum, which are starting to be used for mainframe application development—or to the new, hybrid applications and their protocols. These traditional tools require too much manual effort to adapt, exposing dangerous gaps. Development teams that rely on these tools may take too much time and budget to develop hybrid applications. Application errors and outages may be more frequent, and rollbacks and audits much more time-consuming and labor-intensive.

Fortunately, properly automated mainframe SCCM processes can successfully overcome most of these challenges. Also fortunately, some mainframe SCCM automation technology has continued to evolve along with the mainframe. This modern technology, the processes it automates, and the best practices it enforces can help you lead your application modernization effort.

Modern mainframe software configuration and change management (SCCM) can extend the rigor and stability of the mainframe to hybrid application development and help development organizations scale to the opportunity.

An Ally in Modernizing Mainframe Application Development

Modern mainframe SCCM can be your ally in modern mainframe application development. When intelligently and extensively automated, mainframe SCCM can automatically control every change to application code across the new mainframe development environment, thereby reducing risk and guaranteeing the integrity of production systems. It can extend the rigor and stability of the mainframe to hybrid application development and help development organizations scale to the opportunity.

Modern mainframe SCCM can connect and unify the two developer camps—mainframe and distributed. It will help them collaborate without changing the way each normally works and make both more productive and predictable in their output. Meanwhile, the system polices and guarantees application integrity, auditability, and quality. It can also replicate the smooth handoffs common in traditional mainframe programming environments.

The system achieves these things by intelligently automating software configuration management and software changes across the entire application lifecycle—from development through release management. A modern mainframe SCCM system:

- Captures and codifies mainframe tribal knowledge so it can unobtrusively enforce rules for all developers working with mainframe code or applications.
- Automatically applies mainframe SCCM best practices across the extended development organization. For example, you could develop a policy that prevents a certain type of code change, or a policy that allows a certain change to be made but sends an automatic notification to the project leader.

Modern mainframe SCCM lets you bring the best of both worlds to mainframe application development: the legendary rigor and stability of the mainframe and the flexibility of modern programming languages.

- Governs off-platform developers developing for the mainframe, including Java and C++ developers, reducing the chance of errors upstream in the development cycle.
- Creates and centralizes management information so that it can be used to improve application auditability, process quality, and developer productivity across the application lifecycle. For example, a modern mainframe SCCM system will automatically write everything it does to a log, so there are never any surprises. Rollbacks and audits become much easier, even with greater application complexity.

Modern mainframe SCCM uses native IBM standards, newer standards like Java and XML Web services, frameworks, and open APIs to create open, flexible development environments.

Modern mainframe SCCM includes full native support for Java, HFS, XML, and other technologies in the IBM mainframe application architecture. This means that everything you have been able to do with COBOL and PL/1 working in a 3270 programming environment, you can now do with Java working in Eclipse. And everything you have been able to do with Java working in Eclipse, you can now do on the mainframe. Offplatform developers working in Java can write for the mainframe without knowing the nuances of the mainframe. And the resulting code just works, because the system—not the people—is continually double-checking the connections, tracking the changes, and validating code integrity.

Modern SCCM includes integrated release management. It extends discipline and scalability to your release management process—whether your developers are charged with directly deploying code into production, or you deploy through an application release management team.

Through approaches like those above, modern mainframe SCCM lets you bring the best of both worlds to mainframe application development: the legendary rigor and stability of the mainframe and the flexibility of Java and other modern programming languages.

For example, the customer mentioned earlier in this paper uses modern mainframe SCCM to enable the continuous deployment model for its hybrid applications (Java front-end and DB2 back-end). This customer uses Micro Focus® ChangeMan ZMF and WebSphere to manage the continuous deployment of the JAR (Java Archive) and WAR (Web Application Archive) files that support its applications.

You can further streamline and strengthen modern mainframe application development by taking advantage of advanced concepts such as software packaging. The package concept, pioneered by Serena Software (now Micro Focus), automatically keeps software changes concurrent across the application lifecycle, by managing everything as a package instead of as pieces. As changes are made, package technology automatically updates the relationships between components. Development scales more easily, even as applications become more complex and distributed across different platforms and devices.

Intelligent automation—automation with best practices built in—can automate the mechanics of communication, configuration management, release management, auditing, and other tasks. By automating the mechanics of these tasks, you can free developers to do what they do best: write great code. You can automatically spot errors and often correct them without human intervention. So you can get new applications or application enhancements—even complex, hybrid ones—into production faster, at less cost and with less risk.

Open APIs let you take a more business-focused approach to managing application development.

For example, one company integrated its mainframe SCCM system with its time management system. The system automates chargebacks to the business units for work done on various development assets, without requiring developers to fill out time sheets.

Another company, a leading international bank, wrote a reporting suite that lets its senior managers check on the progress of approvals by individual developer.

Yet another company, a large organization providing payroll services to thousands of clients, uses modern mainframe SCCM to keep changes in sync across the multiple instances of its payroll application, which are used to support different clients. When important changes happen to key components, the system automatically notifies the clients and key stakeholders via email.

Most importantly, modern SCCM creates the discipline and processes that scale with your business as it grows and changes. It makes modern mainframe application development much more predictable, scalable, traceable, and auditable.

Modern SCCM creates the discipline and processes that scale with your business as it grows and changes. It makes modern mainframe application development much more predictable, scalable, traceable, and auditable.

You can take advantage of unique times and new technologies to examine your current processes, upgrade or throw out outmoded ones, and deploy new, more effective ones.

A Golden Opportunity

If you are in charge of mainframe application development, you have a golden opportunity to bring more applications under the control of the mainframe and its legendary rigor. You can use modern mainframe SCCM to improve the way you manage, change, and deploy enterprise applications across the application lifecycle.

You can take advantage of unique times and new technologies to examine your current processes, upgrade or throw out outmoded ones, and deploy new, more effective ones. There's never been a better time than now.

Micro Focus
UK Headquarters
United Kingdom
+44 (0) 1635 565200

U.S. Headquarters
Rockville, Maryland
301 838 5000
877 772 4450

Additional contact information and office locations:
www.microfocus.com

www.microfocus.com