IBM Continuous Integration Solution for System z

Deliver mainframe applications with greater efficiency and at lower cost

Highlights

- Enhance the build process to help validate code—check-in, build, deploy and test
- Automate the development process to provide more efficiency and rapid feedback
- Identify defects earlier in the development process with less effort—and at a lower cost

Poor software quality costs organizations *billions* every year.[1] Furthermore, projects cancelled due to poor quality are 15 percent *more* costly than successful projects of the same size and type.[2] How does this happen?

- Poor visibility into the overall quality of your application codebase
- Manual test and delivery processes slow release schedules and lead to insufficient testing coverage
- Shared test environments and data combined with overlapping release schedules impede innovation, slow code delivery, and possibly lead to inaccurate results.
- Coordination of test environment changes cause bottlenecks, delays and even more overhead.

There is no easy answer. But there *is* an efficient, cost-effective one. The IBM Continuous Integration Solution for System z® can help:

- Automate mainframe and multiplatform application interface testing.
- Determine the status of cross-platform testing progress and quality—in real time.
- Reduce mainframe test MIPS consumption, lowering the cost of testing using an economical System z test environment
- Easily provision and manage a z/OS® test environment.

Mainframe testing and delivery issues

Distributed and mainframe organizations must build and test applications more efficiently and economically. Organizations need to deliver end-to-end applications quickly. This means development teams must implement code changes as quickly and easily as possible, get the testing completed, and the application deployed swiftly. However, based on prevailing mainframe delivery methods, this process can take weeks or even months—just to make simple module changes.

Consider the current process. Developers change code, check it in, build, promote to quality assurance (QA) and go through formal testing and often, environmental configuration changes applied via separate teams. Then, should anything break, it is all fed back to the developers to repeat the cycle. Considering all the steps and responsibilities; especially for newer programmers, it can be a lot to learn. Figure 1 illustrates a typical mainframe development process.

What is Continuous Integration?

Continuous integration is a process that consists of continuously compiling, inspecting, deploying, and testing source code changes. In many continuous integration environments, this means running a new build whenever code within a source code management repository changes. The benefit of continuous integration is simple. Assembling and testing software early and often greatly increases the likelihood that defects can be detected earlier in the development cycle, when they can be more easily managed, and reduces the number of late-stage integration defects which are the cause of many development delays, increased cost, and much rework.[3] Figure 2 shows how the development process can be made more efficient by using a continuous integration solution.

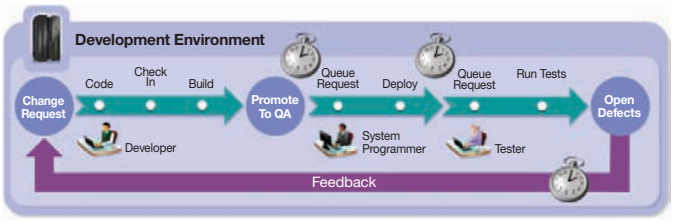

Figure 1. Mainframe development processes can take too much time and can be complicated and costly.

How does continuous integration apply to the mainframe environment?

Continuous integration is a software development practice, not unique for the mainframe, promoting principles which encourage faster delivery and higher code quality. There exists a plethora of industry literature and examples around the practices available via reference books or a simple Internet search. There are 10 high-level practices associated with continuous integration:

- **Maintain a code repository**, to allow developers access to specific versions of code.
- **Automate the build**, to minimize manual handoff and manual steps.
- **Make the build self-testing**, to validate code changes quickly and detect regressions without impacting other developers.
- **Commit to a baseline every day**, to minimize the amount of change to be tested.
- **Commit and build every change**, to ensure that the entire codebase can compile and integrate cleanly.
- **Make the build fast and repeatable** so that feedback is available as soon as possible.

- **Test in a clone of production**, to provide accurate test results with consistent data and middleware configurations.
- **Simplify access to the latest deliverable**, to make re-creation of the test environment easier.
- **Publicize the build status**, to give visibility to any regressions injected into the system via code changes.
- **Automate deployment**, to move code to the next stage in the delivery process once testing is complete.

Organizations do not have to adopt *all* of these practices at once to gain benefits. Each can be adopted incrementally. For example, most mainframe environments already have a code repository and automate the build for stability and consistency. Most already automate some form of deployments and promotion, as well. However, most *don't* have a self-testing build. This is where IBM focuses its continuous integration solution specifically for the mainframe environment. IBM can help make the process as fast and easy as possible—thereby addressing one of the key challenges in today's environment.

The IBM Continuous Integration Solution for System z® (CIz) consists of four integrated products:

- IBM® Rational® Team Concert (RTC)
- IBM Rational Quality Manager (RQM)
- IBM Rational Development and Test Environment for System z (RD&T)
- IBM Rational Test Workbench (RTW)

IBM integrates the capabilities of these four products, optimizing them to implement the continuous integration process. But you don't have to adopt all four to get started. These capabilities can be adopted incrementally, or integrated into your existing software delivery lifecycle tools, to aid in building a continuous integration solution over time and at your own pace.

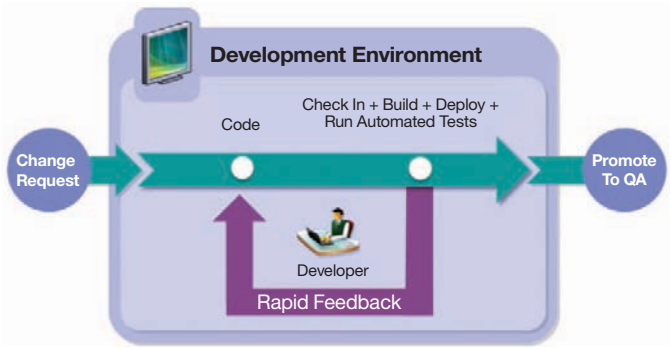

Figure 2. With the IBM Continuous Integration Solution for System z, mainframe development processes are simpler, faster, more efficient and less prone to errors.

For example, if your developers use CA Endevor® to manage source code instead of Rational Team Concert™, you can build code in Endevor and then deploy load modules to RD&T, or a System z logical partition (LPAR). Incrementally adopting capabilities allows your organization to build up a continuous integration system with minimal changes to your existing development process. This helps you realize the value of continuous integration at your pace and budget.

Benefits of a continuous integration solution

Continuous integration helps make the development process easier, more efficient and more cost effective. Faster defect detection via automation, better real time insight, and lower cost test environments are three specific ways to achieve these goals. Let's examine them in more detail.

"With CICS on RD&T, project architects or developers can try changes themselves, in real time. When fully tested, a request is submitted with correct configuration parameters to be implemented on the mainframe. ***This saved the development team weeks.***"

Faster defect detection via automation

On the mainframe, when code is changed, developers might simply run a few manual tests and think the code works. Once it is passed to the test team, which more often than not operates manually, testers might have a thousand pages of test scripts to go through, page-by-page, to complete and validate everything. CIz makes executing these suites of quality assessments easier and more automated.

CIz can help perform all the tests automatically, over time, in a mainframe context. Furthermore, developers don't have to recompile and assemble the entire application to run the tests. Developers can execute the process component-by-component or application-by-application, only updating the modules needing change. You test where you see value and where you're going to get the best quality or performance improvement.

RTC (for build, deploy, and configure) and RTW (for test execution) are important components for automating a continuous integration strategy. RTC detects code check-ins, builds and subsequently deploys load modules, then configures the System z middleware inside a RD&T environment or an existing System z mainframe LPAR. Once modules are deployed and configured, RTW runs automated tests against RD&T or System z LPAR and marks the automated tests as "Pass" or "Fail." Test results are then linked to RTC where the build record directly exposes the test results, providing an automated view into the quality of an individual code change.

With a greater degree of automation, the test cycle can be completed more quickly and code defects detected sooner in the development process, with every code change linked to a quality assessment of that change. RTC helps build more reliable software using an all-in-one development environment for teams—including agile, formal and hybrid planning and reporting—all on a common platform.

To summarize the benefits of RTC:

- **Automated process enforcement**—Controlling the automated check-in, build, and test process
- **Improve communication**—By "smart linking" tasks, code changes, builds, tests, and releases as you work.
- **Live dashboards**—Live, web-based dashboards keep everyone informed of the status of development projects, allowing developers to write code instead of status reports.
- **Empower developers**—Fix errors, juggle tasks, patch in seconds, easily add or remove features.
- **Open platform**—Runs anywhere (x86, System z, Power); Works with your existing tools and data for incremental adoption.

RTW works with RTC to deliver end-to-end functional, regression, load and integration testing to help you execute the application testing process. When you automatically test code changes, defects and regressions are detected quickly with higher degrees of transparency, improving the quality of code being promoted and speeding application delivery.

As teams gather results from the automated tests, you get better risk assessment, better visibility into the results of changes, and less time spent on the manual work of running through a huge number of manual test scripts. This means more time for teams to run exploratory testing and more thoroughly search for additional quality improvements, instead of simply asking, "Did this change break what used to work?"

To summarize its capabilities, RTW helps:

- Simplify the creation of functional and regressions test with storyboard testing, combining natural language test narrative with visual editing through application screen captures.
- Quickly develop complex performance test scenarios with minimal effort using "script-less," visual, performance test and workload models.
- Deliver earlier, end-to-end continuous integration testing across hardware, software, and cloud-based dependencies with support for over 70 technologies and protocols.
- Automate testing and significantly reduce test cycle times by moving testing earlier in the development lifecycle to deliver faster, with improved quality and reduced risk.

Continuous integration helps enable a fast, repeatable, automatic development process and shortens the quality feedback loop from weeks to hours. You get a higher level of confidence and raise the quality of new applications with less effort, at lower cost.

Better real-time insight into application quality and trends

RQM aids in managing the scope of an automated test plan between developers and testers, as well as tracking and reporting the results of tests over time. RQM helps developers and testers evaluate different levels of code and move changes through the system in a controlled, measured, and governed manner. The tracking and reporting aspects facilitate deeper analysis into quality trends, such as which components have a high degree of regressions, or if a release is on schedule to meet a target date for availabily.

RQM enables collaboration and improves productivity across the quality lifecycle, allowing teams to share information easily, use automation to accelerate project schedules, and report on project metrics for informed release decisions.

RQM can help:

- Proactively manage risk, prioritizing the features and functions to be tested based on their significance and the likelihood or impact of failure.
- Assess readiness for delivery of the application with advanced reporting capabilities that address the concerns and needs of quality managers, business analysts and release management.
- Provide at-a-glance access, via user-defined dashboards, to the information most critical to your specific role.
- Better support geographically distributed team communication in context with features such as event feeds, integrated chat and automated traceability.

Lower the cost of System z application testing

At this point, a mainframe developer might ask, "How can running tests on every code change be feasible on my mainframe? I don't have the MIPS to do that!" On the mainframe, more often than not, there is one LPAR where development and testing resides and all developers and testers work with a defined (and constrained) number of MIPS. The single development environment must be scheduled, allocated and setup for each work effort to ensure that different teams do not impact one another during their test cycle, which takes time and limits flexibility.

Using a shared LPAR, there is always some risk that modifying the middleware or test data might affect other teams later in the development cycle, adding additional complexity to running automated tests. If a single row or column of data changes, it may affect other teams in unpredictable ways as automated tests execute concurrently. While operationally low cost, the shared environment can complicate, lengthen and delay testing schedules. A tremendous amount of coordination is required so that every team is ready for each and every change. All project teams must be aligned as changes occur. This is where sharing resources and overlapping schedules can become problematic. Ideally, individual teams would access dedicated test systems, to limit contention and run automated tests, but dedicate the resources in a way which does not dramatically increase development MIPS.

RD&T offers a possible avenue to run large numbers of automated tests in an ad hoc manner. RD&T creates a z/OS test environment on an x86 server, allowing z/OS automated tests to be run without increasing MIPS on the mainframe environment. In fact, some MIPS *reduction* is possible using this solution. RD&T provides development teams their own System z environment. Teams have access to z/OS, CICS®, IMS™, DB2®, WebSphere® Application Server, and MQ. They have modern versions of the COBOL, PL/I, and C++ compilers, all the batch utilities, JES2 and JES3, even an assembler and debugger.

Allocating every team a dedicated environment allows teams to create and control their own set of automated tests and test data. RD&T serves as a team's "personal LPAR." They can change code, build and test as much as needed without impacting the shared development LPAR. Code is tested *when* you want, *where* you want. Validating code changes becomes easier, without increasing MIPS.

A large financial institution created individual development environments for each of their teams, separate from their production environment. Any project team, including outsourced or contract teams, can use their own environment to develop and test projects faster—without worrying about breaking other parts of the system, or outside parties violating security rules by accessing sensitive client data.

To summarize its capabilities, RD&T offers:

- A small-scale z/OS environment on an x86 PC or server, for a wide range of mainframe development or test teams
- The ability to build and test new IBM System z applications virtually anytime, without requiring a scheduled mainframe environment
- A fully functional set of IBM middleware such as CICS, IMS, DB2, MQ, JES, COBOL, PL/I, and Assembler, allows developers create new applications utilizing leading edge z/OS features
- An standalone test environment which can be operated without impacting shared mainframe environments or processes
- A low-cost environment for everyday development use which can free up mainframe development MIPS for additional production capacity

Additional benefits

Continuous integration also helps with risk assessment around changes, such as version-to-version upgrades. For example, developers can prototype compiling using a new version of the compiler, different compiler options, or an updated version of middleware, such as CICS, verifying the code continues to work as expected. This allows developers to assure the team that the upgrade works with the new compiler, options, or middleware officially installed on the real mainframe environment. Additionally, by performing the testing in RD&T instead of on the mainframe, you defer middleware license charge increases associated with upgrading until the upgrade assessment completes. Organizations can reallocate the spare development capacity and budget to other uses, such as additional performance testing, additional capacity testing, or additional production needs.

By providing a RD&T to each development team, you can reduce some of the delays that occur when coordinating all the teams. For example, an individual developer can investigate compiling using the COBOL 4.2 compiler in isolation, without impacting the shared mainframe configuration. Meanwhile, everyone else uses the production-certified COBOL 3.4 version. It enables you to organize *how* tests are run—for more flexibility and quicker delivery—across multiple project schedules.

RTC coordinates the individual environments controlling promotion from each dedicated environment to the mainframe. By reusing some tests from each team, RTC can verify that changes integrate correctly, or else rollback the promotion. All data, code, and environment changes from individual teams are forwarded to the mainframe for further system testing and production.

Conclusion

What should you do next? Why wait? Make a commitment to develop an adoption path to continuous integration. Consider amending your mainframe development process with access to an isolated environment. If you need to improve your development process and shorten the development cycle while controlling costs; start small and adopt incrementally. You could choose to start with Rational Development and Test Environment for System z to lower testing costs and create an environment to automate your testing process, making it more isolated and efficient. But that is just one possibility. You might begin testing using Rational Test Workbench to automate tests, drive better productivity and complete faster assessments of application quality. Or, you could use Rational Quality Manager to further manage and inspect the quality and efficiency of your development deliverables.

The table (See Table 1.) can help you determine which product(s) address your most immediate need.

IBM product	Feature	Benefit
IBM Rational Team Concert	Provides lean, collaborative lifecycle management and build automation including deployment.	• Link code changes to specific builds and work items for transparent access to why changes occur • Automated build, deploy, and testing reduces manual effort and adds repeatability and stability to the development process
IBM Rational Quality Manager	Test planning and monitoring	• Improve application quality and monitor progress of testing. • Produce real-time reports of application health and trends over the release cycle.
IBM Rational Test Workbench	Automates and drives testing of mainframe application interfaces, including Web Services, CICS Transaction Gateway applications, or IMS Connect applications.	• Reduce manual effort when validating code changes. • Detect regressions more easily.
IBM Rational Development and Test Environment for System z	Economical z/OS test environment running on x86 PCs to reduce the development and test load on production mainframes.	• Lower the cost of application testing by using platforms with lower quality of service. • Reduce mainframe development MIPS on production machines. • Create team-specific test environments which are unaffected by changes made by other teams.

Table 1. IBM Rational products that comprise the IBM Continuous Integration Solution for System z. Use this table to identify which IBM product matches the part of your development process that could benefit most from using the IBM Continuous Integration Solution for System z.

There are several other steps you can take to get started today. You can:

- Visit Jazz.net to try RTC or RQM, at no charge, for 90 days.
- Contact your IBM Rational seller for a demo of the RD&T technology.
- Contact your IBM Rational seller for a proof of technology using the IBM Continuous Integration Solution for System z.

IBM is ready to support your continuous integration efforts with more than software. IBM Quick Start services are also available.

Why IBM?

IBM has unmatched experience and expertise in the mainframe environment. IBM offers the only cross platform solution, the only z/OS® continuous integration solution and the only z/OS solution with demonstrable ROI. IBM integrates ALM with test traceability and real time reporting for System z. It uses similar configuration, skills and processes as existing software development lifecycles (SDLC). It even uses the same scripts, reports and procedures as code migrates from development to test.

For more information

To learn more about the Continuous Integration Solution for System z, please contact your IBM representative or IBM Business Partner, or visit the following websites: **ibm.com**/software/rational/integrated/continuous-integration-z/ or **ibm.com**/software/rational/solutions/em/systems/z/

Additionally, IBM Global Financing can help you acquire the software capabilities that your business needs in the most cost-effective and strategic way possible. We'll partner with credit-qualified clients to customize a financing solution to suit your business and development goals, enable effective cash management, and improve your total cost of ownership. Fund your critical IT investment and propel your business forward with IBM Global Financing. For more information, visit: **ibm.com**/financing

IBM Corporation
Software Group
Route 100
Somers, NY 10589

Produced in the United States of America
December 2012

[1] Capers Jones report, 2011 – Based on an analysis of 675 companies, 35 government/military groups, 13,500 projects, 50-75 new projects/month, 24 countries and 15 lawsuits

[2] Paul D. Nielsen, CEO, Software Engineering Institute, Carnegie Mellon University

[3] Andrew Glover, President, Stelligent Incorporated, *Spot defects early with Continuous Integration*

RAS14107-USEN-00