IBM Software
White paper

Meeting the Challenge of Variable Speed IT

Agile development solutions for IBM z Systems

Contents

In today's era of digital disruption empowered by web, mobile, and cloud, it's imperative for enterprise organizations to drive faster innovation while ensuring the stability of core business systems. While innovative systems of engagement (SoE) demand speed, agility and experimentation, existing systems of record (SoR) require similar attributes—with additional and uncompromising requirements for governance and predictability. There is no need to deliver daily updates to production for the SoR if there is no business value to be gained.

However, if your new mobile app needs a critical change in a back-end transactional system, that system of record must be agile enough to provide those services with speed. The issue lies in what is known as Variable Speed IT, in which each element of IT moves at the speed required by the business to meet its goals.

So, how do we transform current environments to meet the demands for faster innovation?

We are all hearing that cultural change is the biggest and most important challenge facing mainframe organizations today. In order for IBM® z Systems,™ and more specifically IBM z/OS,® to be seen as the most efficient System of Record this cultural transformation must take place. No longer can organizations leave the z/OS development team with the same processes, procedures, and tools they have used for the last 30 years.

Today, DevOps transformation is one way of handling this cultural change, to have all teams across the organization come together with the single goal of delivery. This transformation helps break down the silos by creating cross-functional teams to deliver the required business function. These cross-functional teams need to include everyone—from those involved with the mainframe, to those involved in mobile, development, test and operations.

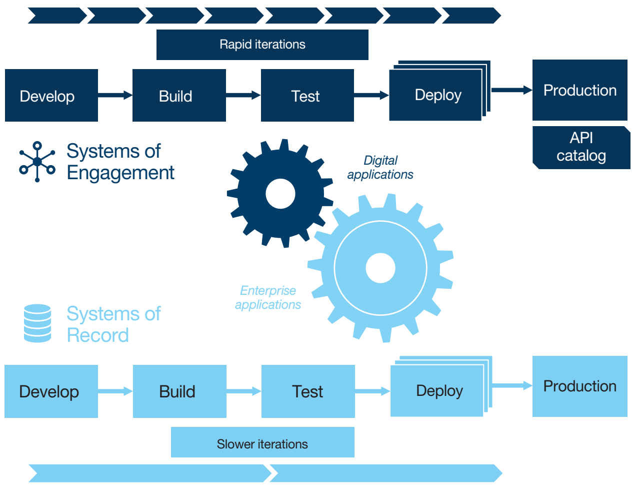

Figure 1: Systems of Record and Systems of Engagement move at variable speeds

In this paper, we'll discuss several solutions that can be incrementally added to modernize enterprise application delivery, or implemented as a group of solutions to align and synchronize collaborative development that, by design, works at variable speeds to deliver business value.

There are many different ways to look at these options for transformation; for the purposes of this paper we have started closest to the metal first. In general, organizations should start with the biggest business need for the transformation. However, some options build on others, and some require more transformation for adoption than others. We'll address:

- Compiler optimization to easily realize productivity gains
- Integrated Development Environment (IDE) to speed delivery of quality code
- Tools to boost the efficiency of problem analysis

Modern developers trust the heavy lifting to modern compilers.

The traditional approach to boosting hardware performance by driving up clock speed is running out of steam; single-threaded performance will not automatically increase as much as it previously did.

Modern developers are turning to the latest versions of their compilers, which can greatly improve the performance of Systems of Record applications, while providing developers with new programming features needed to modernize these back-office applications to integrate with new web, mobile and cloud SoE applications.

Going forward, software performance tuning will play an increased role in delivering performance. As modern IBM z Systems servers become more complex with the introduction of new features, such as decimal floating point, it's becoming increasingly difficult for application developers to simultaneously handle business logic and application performance optimization.

IBM compilers on z Systems are specially designed to allow applications to take advantage of all hardware features provided by IBM z processors that were known at that point in time. Therefore, applications that were compiled a long time ago cannot take advantage of new z/Architecture®.

The operating cost for running these applications could be reduced if they were optimized for the current deployment platform. By periodically upgrading compilers, and selectively recompiling CPU intensive or frequently used applications, organizations can leverage leading-edge optimization technologies to reduce CPU utilization and increase the return on z hardware investment.

Besides the z hardware, applications are also changing. Organizations are rapidly building and expanding new capabilities that tie back-office SoR applications to support front-office SoE applications.

Upgrading compilers can improve the performance of Systems of Record applications, while providing developers with new programming features to modernize applications that are critical to their business. Modernizing applications to work with new infrastructures promotes reuse and extension of proven software assets, allowing organizations to deliver new enhancements quicker, with lower risk and cost.

Furthermore, as an organization adopts the DevOps practice, with automated testing and deployment processes, selectively re-compiling to improve the performance of CPU intensive applications is virtually free and the reward would be savings in operating cost.

The benefits of an Integrated Development Environment for enterprise developers

Another way to improve the agility of SoR development is by adopting modern software development practices, which amplify productivity, quality and responsiveness for most businesses. We've seen that for many organizations, their mainframe development still struggles to keep up; the main inhibitor to success is the development environment being used. Modernizing the development tools is often the first change to support the adoption of new practices.

Five key issues need to be addressed for any development environment to fully support a productive business.

1. *Assistance*—to enable developers to complete tasks
2. *Automation*—to accelerate tasks and remove manual errors cost effectively
3. *In-context capabilities*—core information and functionality needs to be available when and where required switching tools and environments slows development
4. *Integration*—additional functionality and tools should be able to be integrated into the environment (configuration management tools, etc.)
5. *Adoption*—adoption by new users needs to be as fast and as painless as possible.

Assistance

The most productive development environments are able to assist developers in many ways.

First, they operate with knowledge of the technologies being used. Capabilities are designed for the languages being used like COBOL, PL/I, C++, assembler and Java and for the ways they are used in batch, IBM CICS,® IBM IMS™ and IBM DB2.® Basic capabilities, like real-time syntax checking, reduce typing errors and eliminate many unnecessary compilation attempts. More advanced capabilities like application structural analysis, quality analysis and source level debugging optimize the speed and quality of the analysis and debugging phases.

Furthermore, a collaborative lifecycle environment will manage a requirement or change from idea through all phases of development and testing, to promotion and production; this is especially useful for multi-platform application management.

Having the change directly linked to the source allows developers to easily see why other changes in the code were done the way they were. They can also easily see any design documentation or related information to the specific area of code they are working on.

Automation

Even simple process changes bring complexity to the development process, such as requiring manual code reviews. Tools are not strictly necessary to implement code reviews and to see the quality benefits, but there is potential for tremendous overhead. A typical cycle includes the following:

1. Generation of quality metrics
2. Code reviewers assessment
3. Development time to fix issues
4. Repeat until passed

With automation and in-context assistance, quality metrics can be generated as a developer works so they know, before review, whether they have met the required standards. Human review can concentrate on implementation and review failures can be minimized.

Automation also includes automating the build to include the tests, generation of quality metrics etc., as well as the automated deployment. A fully automated integrated development environment helps speed the development process and gets the information back to the developer as quickly as possible.

In-context capabilities

Green screen and legacy development environments really struggle with in-context capabilities. The challenge is that they are restricted to single active areas, with limited space, and only basic display capabilities; this means it is hard to display information in a convenient way, and switching between functions can be laborious and severely limiting.

As a result enterprise developers usually operate in a very linear sequential manner and find it hard to rapidly switch tasks—putting them under tremendous pressure when agile and lean processes are introduced, since the environment denies them the flexibility they need.

Even using emulators to give multiple windows does little to alleviate the restrictions. Modern IDEs operating in multi-window environments, like Eclipse, immediately allow all relevant information and functionality to be available on-demand and in context. Simply allowing developers the luxury of having all relevant components of an application flow on one pane eliminates tedious back- and-forth when developing.

Integration
Switching between tools is inefficient and provides opportunities to make mistakes. To ensure processes run efficiently as possible, IDEs need to integrate external functionality directly into their environment. Typical areas where this integration is highly desirable are:

- File and data management
- Debugging
- Software configuration and change management
- Source control management

Through a fully integrated development environment including change management and source control management, developers are more likely to properly manage the changes, and update their activities to keep plans current.

Adoption
Enterprise development is suffering from an ageing development community and organizations are constantly striving to bring new developers on board. They face two hurdles: the technologies are unfamiliar to most of the available developers, and the environment is not particularly attractive (current, marketable, sexy).

Currently, the best way to overcome these challenges is to use an Eclipse-based IDE with an assisted development environment.

Eclipse is used in schools and colleges around the world, so new developers are already familiar with it. Built-in assistance for the languages and technology, such as COBOL, PL/I, DB2, means that developers can become productive significantly faster than when introduced to green screen development.

IDEs for enterprise development are now able to offer tools which are comparable and in some cases even more powerful than on other platforms. This now makes a very attractive package to new developers because they can add a highly marketable skill set to their existing knowledge base.

Tools to boost the efficiency of problem analysis
Now we'll look at some tools that can help developers with problem analysis as well as the DevOps practice of minimizing handoffs and maximizing flow. The goal is to allow development and quality assurance teams to see early on how the environment will behave, and to make course corrections as appropriate.

Such tools can be categorized as:

Application performance analysis: Helps to understand resource consumption by systems and applications. Such tools help end users identify resource constraints and areas with excessive resource consumption. The observation leads to a specific set of actions to improve overall performance, which may result in reduction of computing resource usages, reduction of response time, and identification of erroneous application logic, among others.

Interactive debugging: Enables users to step through application logic to follow and understand what's been executed while allowing users to examine associated resources at a specific point during the execution of a program. It not only allows interactive debugging for a user, but it is typically used as a foundation for enabling quality assurance tools; for example, code coverage analysis tools which enable testers to understand coverage of a regression test suite.

Failure analysis: Applications and systems fail despite significant advancement in technology and rigorous quality improvement practices. Failure analysis tools, which aid in identifying the root cause of failures down to application source code level, play an increased role in the era of mobile computing as lengthy down time typically means lost opportunities for businesses.

Integrated problem analysis tools help you deliver at the speed your business requires. These tools are designed to be used in production systems as well as development and test systems; problem analysis requirements may demand intervention and observation in the production environment due to the differences in size of the production vs any test environment. This is essential for ensuring application developers and system programmers are provided with the right level of problem analysis information. For example, let's look at post mortem debugging.

Failures occur in production systems when least expected, even with rigorous quality assurance practices and procedures. Sometimes these failures are difficult or impossible to reproduce in a test environment. Capturing relevant facts around the failing application is the first step in reducing the downtime; however, in instances when the root cause cannot be absolutely pinpointed, post mortem debugging is a logical next step.

What other challenges are to be considered? Because z Systems applications may consist of assets acquired over decades of continuous operation, it is inevitable that the assets are diverse and heterogeneous. Some are the result of incremental in-house development and maintenance, while some are purchased from third parties, and others might be the assets acquired during a business merger.

With unmatched backward compatibility and technology advancements of z Systems, it is common to find assets using a wide range of technology available on the platform. For example, applications can be mixture of assets developed in OS/VS COBOL, non-LE Assembler and Java. Therefore it is critical to use integrated tools that provide coverage across all languages and subsystem environments that are most commonly used by z Systems applications.

Conclusion

As discussed in this paper, a transformation is needed for the development and operations of z Systems applications to meet the digital demands being put on today's businesses; one that includes using common tools that support the modern development practices of Agile and Lean IT, as well as provide the transparency and automation required to move at the speed of business today.

So, how do we unhook unnecessary dependencies in large monolithic applications going forward? This can be addressed by the use of modern application programming interfaces (APIs) which provide a set of routines, protocols, and tools for building modern software applications. API management is the process of publishing, promoting and overseeing APIs in a secure, scalable environment. These APIs are created based on the existing applications through a process of transformation from large monolithic applications to independent services. This is more easily achieved with modern tooling providing wizards to help build the interface and additional testing capabilities.

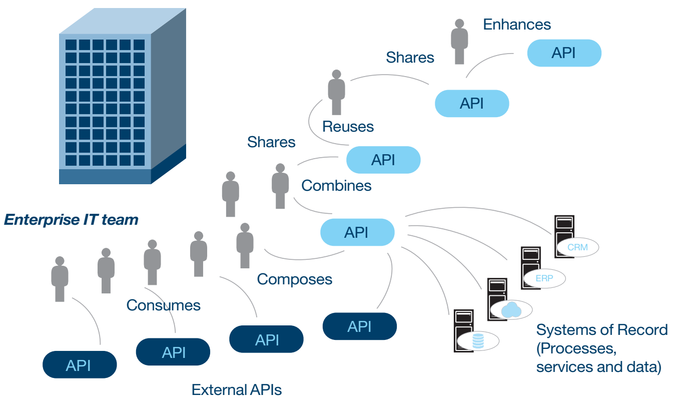

Figure 2: Application Program Interfaces (APIs) power the modern, digital supply chain

It is becoming increasingly important as APIs can define the integration as services connecting SoE and SoR. By using well-defined APIs, organizations can loosely couple SoR and SoE, allowing them to move at their own speed. APIs with proper forward and backward compatibility will be required. The combination of a loose coupling of SoR and SoE, along with service oriented APIs, brings the added advantage of being able to simplify the development and testing process.

With defined APIs as an interface, virtual services can easily be created to allow the SoE to build and test the capabilities without actually having access to the SoR. This helps to insure that when the integration testing is performed, any problems have been identified and fixed earlier. Allowing the SoE and SoR teams to work independently through loose coupling helps develop function faster with fewer system resources and increases the productivity of all the teams by not having to wait on each other.

Enterprises need instrumented and automated coordination to address the challenges of variable speed IT. DevOps provides the gear mesh between traditional systems and new systems of interaction that may have different rates of development, standards of security and user expectations. Using a defined bundle of integrated solutions in a DevOps environment is the “simple machine” way to generate the speed and power needed to deliver business results in the most agile and lean manner in your enterprise environment.

For more information

Enterprise COBOL for z/OS
ibm.com/software/products/en/entecoboforzos

Rational Developer for z Systems
ibm.com/software/products/en/rational-developer-for-z-systems

z/OS Problem Determination Tools
ibm.com/software/awdtools/deployment

KUW12369-USEN-00