Mainframe Modernization

By Philip Russom

Sponsored by

Progress | DataDirect

www.tdwi.org
FOREWORD

NUMBER ONE
Bridge the modernization gap between the mainframe and other systems.

NUMBER TWO
Integrate mainframe data at various right-time speeds.

NUMBER THREE
Move data bidirectionally onto and off of the mainframe.

NUMBER FOUR
Reduce the volume of data coming from the mainframe.

NUMBER FIVE
Process mainframe data as far upstream as possible.

NUMBER SIX
Consider making services the foundation of mainframe modernization.

ABOUT OUR SPONSOR
There is no question that IBM System z mainframes continue to serve a wide range of organizations by providing a secure, high-performance, and scalable computing platform that’s hard to match on other systems. The issue comes when you attempt to extend mainframe data or applications to participate in new business applications on so-called open systems (servers running Linux, UNIX, or Windows) and Web environments (whether Internet, intranet, or extranet).

Three factors have led to the isolation of mainframes: First, most data integration and application integration tools and servers available from vendors today were designed for open systems and Web environments. Until recently, few vendor tools could run natively on mainframes or even support their standard interfaces. Second, it’s taken more than a decade to develop and adopt industry standards for data and application interoperability—a necessity if proprietary mainframe artifacts are to participate outside the mainframe’s monolithic architecture. And third, in many people’s minds, the mainframe is legacy and ripe for replacement, primarily because of its outdated total cost of ownership (TCO) model. But mainframes aren’t going away, because they support massive databases and applications that would be expensive, risky, or impossible to support on open systems and Web environments. Hence, organizations need to invest in modern technologies for mainframe modernization to connect the mainframe with other enterprise platforms.

The best business case for mainframe modernization is that it enables a broad range of modern, time-sensitive business practices, including on-demand performance management, just-in-time inventory, operational business intelligence, and business process integration. Before the mainframe can participate in these business and technology activities, it must be integrated with other enterprise platforms.

Mainframe modernization takes many forms. For many organizations, it’s about providing a more streamlined method for using mainframe information on other platforms. For others, it’s about extending the mainframe to the Web. Some forward-looking organizations are making the mainframe an active participant in service-based composite applications, utilizing Web services standards to support a service-oriented architecture (SOA). Organizations seeking the greatest value from the mainframe must consider all of these factors. This Checklist Report touches on all aspects of mainframe modernization, but focuses primarily on data integration issues.
“Right time” is about providing information within a time frame that’s appropriate for that information. Time frames range from real time (less than a second) to batch processing (typically overnight). The reason for different time frames is that every datum potentially has its own requirements for information delivery speed, depending on how time-sensitive the datum is for the business. Enabling a right-time business response through technology is challenging, because different integration technologies operate at different speeds. So, one of the consequences of right time is that an organization may require multiple mainframe integration technologies when it needs to enable multiple information delivery speeds.

With mainframes, right-time integration has different meanings, applications, and enabling technologies:

- **Real time.** This is the fastest extreme of right time, involving subsecond response time. Real-time integration is appropriate when data needs to be integrated across platforms. This may entail time-sensitive data (typically about a customer’s financial status) or data that must be synchronized across systems (as with changes to a customer’s master record). Since information is usually packaged as messages, enabling technologies include tools that can natively access the mainframe data (in either an asynchronous or synchronous fashion) and make it available to platforms for enterprise application integration (EAI) or message-oriented middleware (MOM).

- **Event driven.** For example, when a customer exceeds his/her credit line (as seen from a mainframe-based credit card approval application), this event (a technical data update) would be captured, filtered, maybe even aggregated with other contextual information, and then delivered in real time to an open-systems application to enhance customer service and so on. The message travels via the real-time technologies just mentioned. If multiple events must be evaluated before a message is sent, complex event processing (CEP) may be required. Since an event can be a transaction or any change in the state of mainframe data, event-driven mainframe integration can also enable changed data capture (CDC).

- **On demand.** This is appropriate to reporting solutions that need relatively fresh data, but can wait several seconds or minutes for a response to a query. As an example, consider performance management as seen in executive dashboards; managers need to refresh the data in their dashboards multiple times daily (even at unpredictable moments) to understand and react to the performance of time-sensitive processes, such as inventory, call center workloads, and unit yield on a manufacturing floor. Structured Query Language (SQL) is usually a requirement for such applications. SQL statements and result sets may be transported via drivers for Open Database Connectivity (ODBC), Java Database Connectivity (JDBC), or ActiveX Data Objects (ADO.NET). Or SQL queries may be expressed by tools for reporting, data analysis, or data federation. In all of these cases, some degree of data mapping (and processing) are required to transform nonrelational data structures (typical of mainframe data) into a relational format (typical of databases on open systems).

- **Batch processing.** This is the slowest extreme of right time, typically involving a 24-hour or longer cycle. Most mainframe data integration involves files, whether data is departing the mainframe or being loaded into it. Most files are generated and processed in overnight batch windows. This makes mainframe integration via files rather slow compared with other options. Yet, the batch processing of files is very useful for large data sets that are not time sensitive. For example, some data warehousing and business intelligence tasks based on historical data work just fine with mainframe data that’s delivered with the high latency of batch. As another example, product catalog data doesn’t change frequently, so synchronizing it across the mainframe and other systems is best done in overnight batches. That being said, files over FTP shouldn’t be an organization’s only data path into and out of the mainframe.

- **Services.** Web services and SOA are arguably two of the most significant integration innovations to hit mainframes in recent years. Most of the examples of right-time information delivery speeds noted here can be handled through a service or a collection of services. As explained at the end of this Checklist Report, the many applications and benefits of services are quickly making services the preferred platform and architecture for mainframe modernization.
Anecdotal evidence suggests that most companies using an IBM System z more often tend to move data off the mainframe rather than onto it. In these cases, a few large applications run natively on the mainframe, taking advantage of its massive capacity and reliability. Data being integrated flows from mainframe applications and databases to support Web or data-centric applications on open platforms, although some business processes would benefit from data flowing in the reverse direction.

- **Closed loop.** From a technology viewpoint, moving data from mainframe databases is relatively easy, involving files generated by simple table or database dumps. However, closing the loop by moving data into a mainframe database is relatively difficult, because the data must almost always go through the application logic of a legacy application. Even so, some companies “write back” time-sensitive data to a mainframe through EAI or MOM, and less often through a customer information control system (CICS) or an ODBC/JDBC/ADO.NET connection. The write-back function has traditionally demanded programmatic integration, especially when the process requires the data to be written back through existing programs or 3270 screens. The ability to expose these as Web services has the potential to greatly reduce the time and costs of development.

- **Data synchronization.** Moving data bidirectionally onto and off of the mainframe is desirable when the business needs up-to-date, complete, and consistent information synchronized on both the host and distributed platform. Regardless of the platform, when a data change occurs, it should be propagated to the other side of the modernization gap. This is the case when synchronizing data about customers, products, and financials for a 360-degree view of these business entities on both sides of the modernization gap.

- **Bidirectional services.** Whether using independent Web services or a unified SOA, most services that integrate data can be designed for bidirectional data movement or deployed in sets of services that constitute a closed loop. Mainframe Web services requester technologies typically allow a mainframe program to make a call to an external application or database to retrieve the information and return it the mainframe program originating the call.

Extractions of mainframe data tend to include more data than any user or application needs. For example, dumps of entire tables or databases are common. The result is a very large volume of extracted mainframe data that a downstream data integration server must process, in search of the data subset it actually needs. The downstream processing of large mainframe data sets threatens the scalability of integration infrastructure; it also slows down information delivery speed. So it behooves you to reduce the volume of extracted mainframe data.

**Extraction via SQL statements** is one of the best ways to reduce the volume of extracted mainframe data. Through SQL, you define exactly the data you need, and you can transform and join extracted data (to a certain degree). This requires support for SQL in the mainframe database, or an additional solution that provides SQL access to the mainframe’s nonrelational data. In both cases, a standards-based API, such as ODBC/JDBC/ADO.NET, is required between the mainframe and distributed applications. Note that typical implementations can be resource intensive, so considerations for capacity constraints should be part of the requirements analysis. When done right, this approach is worth the effort, because, compared with table dumps and legacy routines, SQL delivers less data in much better condition, with less development time and complexity.

**Changed data capture (CDC)** is an even more effective method. CDC automatically captures data as soon as it is created or changed, and provides facilities to deliver it to specific destinations. This completely alleviates the need for data extracts and subsequent loads. Furthermore, CDC can be configured to filter and aggregate data changes into more relevant events, to process changes in batch mode, or to stream changed data continuously for real-time data synchronization. CDC requires an event processing facility that has native access to mainframe databases, event persistence to avoid losing an event, and the ability to deliver the events via a transport protocol such as WebSphere MQ or HTTP.
Native data processing on the mainframe. Ideally, you should process mainframe data natively on the mainframe—to cleanse it, reduce its volume, or otherwise improve it—before passing it to other platforms. After all, data sets drawn from legacy platforms usually have significant quality issues, and data volumes extracted from a mainframe tend to be large. But there are barriers to native processing. Relatively few data integration and data quality tools can operate natively on a mainframe in a cost-effective manner. When all mainframe MIPS are committed or additional tools incur additional costs for capacity, adding a data quality or integration tool isn’t an option.

Specialty engines. The prospects for native data integration and quality processing on a mainframe have gotten much better in recent years, thanks to IBM’s introduction of specialty engines. For example, the IBM Integrated Facility for Linux (IFL) enables many types of enterprise software (including integration and data management tools) that run on certain versions of the Linux operating system to also run in an IBM System z environment. More recently, IBM introduced the IBM System z Integrated Information Processor (zIIP) and the IBM System z Application Assist Processor (zAAP). Specialty engines are acquired through a one-time purchase price and provide processing for defined workloads, namely: IFL for processing within z/Linux, zAAP for Java on z/OS, and zIIP for z/OS processing that meets IBM-defined requirements. Specialty engines are exactly like a mainframe’s General Purpose Processor (GPP), except that specialty engine workloads do not count against mainframe MIPS or MsUs, and the speed at which they run is not restricted. Mainframe integration middleware—when specifically designed to exploit specialty engines in accordance with IBM’s authorized use—can shift loads associated with data integration and application integration to the zIIP and zAAP engines, respectively. These specialty engines provide new possibilities for native data and integration processing on the mainframe—all at a relatively low cost point.

A unified architecture for mainframe integration. If you implement all the recommendations made in this Checklist Report, you could easily end up with a plague of interfaces, possibly designed and managed by separate tools. The result is a spaghetti pot full of point-to-point interfaces that increases costs and complexity. The lack of a unified software architecture—all too common with mainframe integration solutions—has dire consequences. Data standards, developer productivity, performance tuning, administration, integration availability—all these suffer without a sustainable architectural design.

Strategies for mainframe modernization. The challenge of designing an architecture is exacerbated when mainframe modernization has diverse meanings and goals, including integrating nonrelational data with relational databases, extending legacy programs to the Web, capturing database changes to support event-driven integration, and making the mainframe an active participant in service-driven composite applications. All of these strategies are required by organizations seeking the greatest value from the mainframe. All bridge the modernization gap, but with very different tools, techniques, and business goals. The catch is to find a common technology platform and holistic architectural design that enables them all in a modern and cost-effective manner.

Services as the foundation of mainframe modernization. One way to satisfy the need for a single technology platform and architectural design for diverse mainframe modernization strategies is to embrace a services approach. Ironically, this leading-edge technology is ideal for modernizing the ultimate legacy platform: the IBM System z. This requires an SOA layer for the mainframe. But it’s worth the investment, because most of the interface and tool types discussed in this report for data access, integration, and quality can be exposed through services, as can many legacy renewal techniques. SOA incorporates the interoperability benefits of request-driven services and the real-time intelligence of event-driven integration. When SOA is in place elsewhere in the enterprise, the mainframe can seamlessly tie into SOA infrastructure, as well as the integration infrastructure available through SOA. Using wrappers, services can make mainframe application logic available to a variety of applications, including composite applications. And SOA has been proven to support best practices in application architecture and consistent developer standards.
Progress DataDirect.

Progress DataDirect is the software industry’s only comprehensive provider of software for connecting the world’s most critical business applications to data and services, running on any platform, using proven and emerging standards. Developers worldwide depend on Progress DataDirect products to connect their applications to an unparalleled range of data sources using standards-based interfaces such as ODBC, JDBC™ and ADO.NET, XQuery, and SOAP. More than 300 leading independent software vendors and thousands of enterprises rely on Progress DataDirect to simplify and streamline data connectivity for distributed systems and to reduce the complexity of mainframe integration.

Progress DataDirect Shadow is a single, industry-standard architecture that reduces mainframe integration complexity, cost, and risk. Shadow simplifies mainframe modernization, allowing organizations to easily extend their mainframe data, programs, and screens to participate with modern service-oriented architectures (SOA), Web or composite application development, or data-intensive initiatives like business intelligence or data warehousing. Unlike any other mainframe middleware today, Shadow employs a unique, patent-pending technology for holistic exploitation of mainframe specialty engines that reduces up to 99% of the data integration and SOA-related integration processing—all without offloading any IBM or third-party ISV code.

Progress DataDirect is an operating company of Progress Software Corporation (NASDAQ: PRGS). For more information, visit www.datadirect.com.

ABOUT OUR SPONSOR

ABOUT THE AUTHOR

Philip Russom is the senior manager of TDWI Research at The Data Warehousing Institute (TDWI), where he oversees many of TDWI’s research-oriented publications, services, and events. He’s been an industry analyst at Forrester Research, Giga Information Group, and Hurwitz Group, where he researched, wrote, spoke, and consulted about BI issues. Before that, Russom worked in technical and marketing positions for various database vendors.

ABOUT TDWI RESEARCH

TDWI Research provides research and advice for BI professionals worldwide. TDWI Research focuses exclusively on BI/DW issues and teams up with industry practitioners to deliver both broad and deep understanding of the business and technical issues surrounding the deployment of business intelligence and data warehousing solutions. TDWI Research offers reports, commentary, and inquiry services via a worldwide Membership program and provides custom research, benchmarking, and strategic planning services to user and vendor organizations.

ABOUT THE TDWI CHECKLIST REPORT SERIES

TDWI Checklist Reports provide an overview of success factors for a specific project in business intelligence, data warehousing, or a related data management discipline. Companies may use this overview to get organized before beginning a project or to identify goals and areas of improvement for current projects.