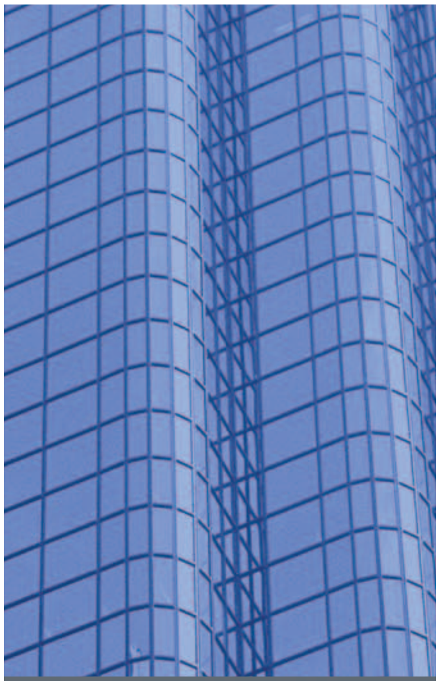

SHADOW®

whitepaper

Occam's Razor Guide to Mainframe Integration

www.neonsys.com

NEON SYSTEMS, INC.

SHADOW®

contents

NEON Systems, Inc. Corporate Headquarters
14100 Southwest Freeway, Suite 500
Sugar Land, TX 77478
Phone: 281.491.4200 Fax: 281.242.3880

NEON Systems New York Regional Sales Office
1002 Washington Street, Suite 111
Hoboken, NJ 07030
Phone: 201.653.8113

NEON Systems Dallas Regional Sales Office
HQ Colonnade Center, 15305 Dallas Parkway, Suite 300
Addison, TX 75001
Phone: 972.387.7440 Fax: 972.715.2099

NEON Systems UK, Ltd.
1 High Street, Windsor, Berkshire SL4 1LD
United Kingdom
Phone: +44.1753.752800 Fax: +44.1753.752818

NEON Systems Japan
2-6-2, Ootemachi, Chiyoda-ku, Tokyo 100-0004
Japan
Phone: +81.0.90.8027.9145 Fax: +81.0.3.6202.9151

Executive Summary

Few organizations today are embarking on significant new application development that runs in the traditional IBM mainframe environment. Most modern applications, designed to improve operational efficiencies and streamline customer, supplier, partner and employee interaction, run within J2EE or .NET environments on distributed UNIX or Windows server systems. However, for the Global 2000 and many government agencies the vast majority of the data that these new applications require is maintained under control of IBM mainframe systems, specifically, legacy database systems such as Adabas, IDMS, IMS/DB and VSAM, or relational database systems such as DB2. In many cases the data itself is meaningless without business rules, buried within legacy CICS or IMS/TM application environments to interpret it.

The reasons for a white paper on mainframe integration is that enabling those new applications to access the mainframe resident information isn't just important; without it, the applications themselves are worthless since they have no corporate information upon which to operate. For some peculiar reason this rather fundamental point has failed to register on the radar screens of corporate architects responsible for building out middleware infrastructure. Significant time is certainly spent on analysis of application servers, messaging systems and Integration Brokers, yet the last mile of integration to the crown jewels of most organizations is relegated to an afterthought. The net effect of this complete absence of strategy, in respect to mainframe integration, has resulted in many failed projects – a complete application being only as strong as its weakest link.

This paper explores the underlying reasons why mainframe integration, if not approached strategically, is almost guaranteed to fail or at best be a long-term drain on both hard and soft finances without providing a service upon which the business can rely. It looks at why the traditional approaches have failed to deliver reliable and cost-effective mainframe integration and suggests a strategic and far simpler alternative, proven to work in the world's largest organizations. The entire piece is set against the reasoning set forth in the scientific precept known as Occam's Razor.

What is Occam's Razor?

Ockham is a small town some 15 miles southwest of central London. Today it is known largely for its commercial airways navigation beacon and the fact that it is a rather convenient cut-through when the London orbital freeway is blocked. Obviously, it wasn't always thus. In the 13th century a chap by the name of William was born in Ockham; although, back then it was spelled Occam. Nothing remarkable in that, however, William, almost unknowingly, defined a scientific precept that has been used in some of man's most important discoveries ever since. From religion to relativity, the precept, known as Occam's Razor, has distilled scientific thinking and provided a pathway to reasoned solutions in complex problems.

In its original form, William of Occam suggested, "*pluraitas non est ponenda sine necessitate.*" In English this translates to "*plurality should not be posited without necessity,*" which is often simplified to say "*that which can be done with fewer, is done in vain with more.*" This has been interpreted down the years in many ways; however, at its core Occam's Razor is understood to mean that if there are several solutions to the same problem, the simplest will be correct. Many of history's most renowned scientists have applied the Razor successfully, and it is now a core control in scientific reasoning.

Its relevance in computing needs little explanation. For many years simplicity has been a strategic directive in enterprise architectures. However, as composite applications, which depend on legacy data and applications, become commonplace on the project plans of most large organizations, Occam's Razor must be applied to legacy integration. Seen by many as an afterthought and worthy of little strategic consideration, mainframe integration implementations have rarely attracted a strategic approach. Consequently, most organizations with a history of linking legacy systems with distributed applications find themselves with multiple fragmented and incompatible systems. Such implementations have huge implications for business agility, costs and application reliability.

Causes of Complexity

Complexity in commercial computing systems emerges from a relatively narrow set of reasons. Many of these reasons exist in the wake of the shift away from the single-vendor, vertically integrated solutions of the 1970s and 1980s. Cost savings from the increased competition brought by a move to a more varied vendor environment have been offset by the increased systems management and consulting costs required to implement systems that are no longer integrated at source. Standards and other initiatives, aimed at making systems from various vendors compatible, are an important step toward mitigating some of these costs, but will certainly not eliminate all of them; the most intractable exist as a fundamental consequence of the modern computing landscape.

Heterogeneity of Vendors

Heterogeneity of vendors is a defining element of today's commercial computing landscape. The vertically integrated computing platform, provided by IBM of old, has given way to various commercial-off-the-shelf (COTS) packages built to address the business processes of various industries, and provide economies of scale by eliminating the costs of duplicate bespoke application development. The net result is that organizations now maintain more significant relationships with suppliers of IT software, hardware and services than ever before. Such vendor diversity makes it difficult for customers to assign responsibility to any single vendor for integration issues. This is problematic at a commercial level, but even more insidious at a technical level.

Multiplicity of Support Organizations

With multiple support organizations from multiple vendors responsible for different elements of an end-to-end service, seeking timely resolution for failures somewhere along the pathway from provider to consumer can be difficult if a failure occurs close to the hand-off point between one technology and another. Such scenarios require the support organizations of the vendors involved to collaborate in search of a resolution. With no way to directly assign ownership, it is difficult for a customer to hold any single vendor responsible. Even the introduction of a service provider layer does little to speed the process. The best solution is to seek the least possible moving parts at inception of the project.

Middleware Distance Between Consumer and Provider

The more separate components, both hardware and software, dedicated to delivery of middleware to link the ultimate consumer of the service and the underlying applications and data required, the more complexity will be experienced. This includes hardware platforms deployed specifically to support middleware products (and the attendant failover provision) and software translation layers for turning data into one format or structure from another.

Systems Management Requirements for End-to-End Coverage

Systems management is a core requirement for any commercial system. This may seem like an obvious statement as systems management exists to ensure that the service to the end-user measures up to expectations. In order to satisfy this objective it is critical that threats to that service, from whatever source, are identified and diagnosed in as timely a manner as possible. Most individual elements of an integrated system will provide their own systems management capability, but rarely will the hand-off between juxtaposed elements be seamless. Tracking a transaction, which flows across multiple software and hardware layers, from end-to-end is a complex task where those layers have nothing technologically in common. The net result, for the service management personnel, is a bank of systems management consoles with loose, often human, systems for tracking transactions as they flow through the entire system.

Cost of Complexity

The only justification to embark on the kind of analysis alluded to herein is cost-efficiency. If complexity has no impact of overall cost, then why bother? I am sure many readers will instinctively believe complex systems are more expensive; however, in any study that suggests alternative strategies it is helpful to detail in precise terms the more common cost implications of complex solutions.

In the following sections, reference is introduced to the mainframe from an integration complexity perspective as a way of illustrating how complexity can have attendant costs.

Hard Costs

Software

The more middleware technologies employed to satisfy an integration requirement, the higher the ongoing software maintenance and license fees will be. For organizations with the kind of complex solution outlined below, such fees can become material. Unfortunately the very nature by which these technologies come to be used often masks the overall costs associated with them. It is often the case that individual projects, faced with a mainframe integration requirement, will purchase their own solution; the costs being defrayed under the cost of the project itself. Consequently, there is rarely a systematic way to determine how much has been spent on mainframe integration across the entire enterprise.

Hardware

Most mainframe integration solutions have significant hardware dependencies in the form of Wintel or UNIX gateway machines. Such architectures have ongoing hardware costs to support workload scaling and, for mission-critical workloads, have gateway duplication to support failover. With a fully installed gateway device running to well over $10,000, for large implementations these costs mount up rapidly. Again, if an organization has approached mainframe integration in a haphazard way, a gateway farm is likely to exist for each discrete mainframe integration scenario. This can add hundreds of thousands of dollars to the software license fees for the mainframe integration requirements of applications with high-availability and throughput needs.

Soft Costs

Although far less easy to track, the soft cost implications of complexity are far more troublesome than the hard costs.

Business Agility

As has always been the case, a competitive business must be agile enough to respond in a timely manner to opportunities and threats from within its chosen market. In order for these responses to be enshrined in automated processes, IT architectures must be developed that can adapt rapidly regardless of the demands placed upon them by business responses to these threats and opportunities. When performing an impact analysis to assess the risk of introducing a new workload, a complex architecture is certain to be more susceptible since it has more moving parts to impact. Also, a complex system has more permutations to analyze, which extends the timeframe for deploying a new system. Furthermore, complex systems have higher risk of failure since there is always a possibility that some combination of effects have not been considered during the impact analysis.

Personnel

Generally speaking, personnel costs increase in line with complexity in support of increased systems management workload. As more hardware platforms and physical interconnections are deployed the demands on hardware oriented management personnel increases. More significantly is the impact on software systems management personnel as the number of middleware technologies increases. Most integration technologies come equipped with their own breed of systems management. Rarely is the instrumentation integrated in a meaningful way with juxtaposed systems management software. The net result is a bewildering array of tools to provide end-to-end visibility for composite applications. It is expecting a lot for such a systems management environment to be managed effectively with staffing levels of simpler systems. With staff costs representing the majority of expenses in any IT organization, complex IT architectures can be seen to have a cascading effect on an overall budget.

Availability

High-availability is a defining characteristic of most mission-critical systems; revenue is impacted when mission-critical systems fail. Mission-critical composite applications are no different. The more moving parts in any end-to-end application, the more scope there is for a service interruption. Given the choice of a single technology for all mainframe integration requirements versus multiple components, with multiple hardware and software layers, it would be hard, from an availability standpoint, to choose the former. Yet that is precisely what many organizations end up deploying, as a strategic approach to mainframe integration is rarely considered.

Emergent Complexity in Mainframe Integration

Few people would admit to designing complexity into a computing architecture from the very beginning. However, when presented with the bald facts of most mainframe integration architectures, it would seem at first blush that the pursuit of complexity must have featured high up on the original design goal. There are some key reasons why mainframe integration, more than most enterprise architectural disciplines, has a tendency toward complexity.

Mainframe Uniqueness – The Provider Perspective

Unlike most commercial computing platforms, mainframes routinely support multiple competing workloads, with multiple implementation options. To be specific, it is commonplace for a single mainframe environment to support the full range of business applications implemented across CICS applications (both screen-oriented and program-oriented), batch programs, relational and non-relational databases, fourth-generation language systems, to name but a few. When this run-time implementation is overlaid with typical integration requirements such as high-volume transactional access to existing programs, data synchronization, SQL access, Web-to-host integration, the technological permutations for integration are mind-boggling. Most mainframe integration implementations are never conceived from the perspective of "mainframe" integration. Rather, integration technology is chosen to satisfy the point requirements identified above. Examples being a data synchronization product to keep a business intelligence data mart in sync with the mainframe IMS database; a CICS screen-scraper product to Web-enable a call-center application; an ODBC driver for DB2 to allow direct data integration for a portal application; and a J2CA interface for IMS to enable an Integration Broker product to run IMS transactions and have processes triggered by IMS events. Each integration requirement is tackled as a separate and distinct project from the next because rarely are each of these needs seen as elements of a holistic mainframe integration project; rather, they are seen as complete integration projects in themselves to support a particular business need. This is largely due to the pressures applied by the non-mainframe application development and systems integration personnel.

Application Development – The Consumer Perspective

Application development and systems integration personnel are accustomed to simple adapter implementations to satisfy the full-range of integration needs for any given business system. Integration technologies for SAP, Siebel or Peoplesoft ERP implementations, which manifest themselves as "Adapters" satisfy all integration needs from within a single adapter instance. The simple adapter metaphor is the expected paradigm for integration of a single business system. When an integration project needs access to a mainframe asset such as a CICS application or IDMS database, the expectation is for a low-cost, single-purpose adapter to satisfy the needs of the project only. Depending on the procurement processes within each organization it may be that the mainframe adapter, to satisfy this particular business application, is acquired by the project team itself. Even if it is considered a central services item, it is likely that project ROI pressures will mean that a simple, single-purpose technology is chosen that can satisfy only the needs of the project in question.

When you combine these two perspectives it is almost inevitable that each organization's mainframe integration technological landscape resembles the quintessential spaghetti diagram (see below), with incompatible, piecemeal solutions deployed to support each project as a separate need.

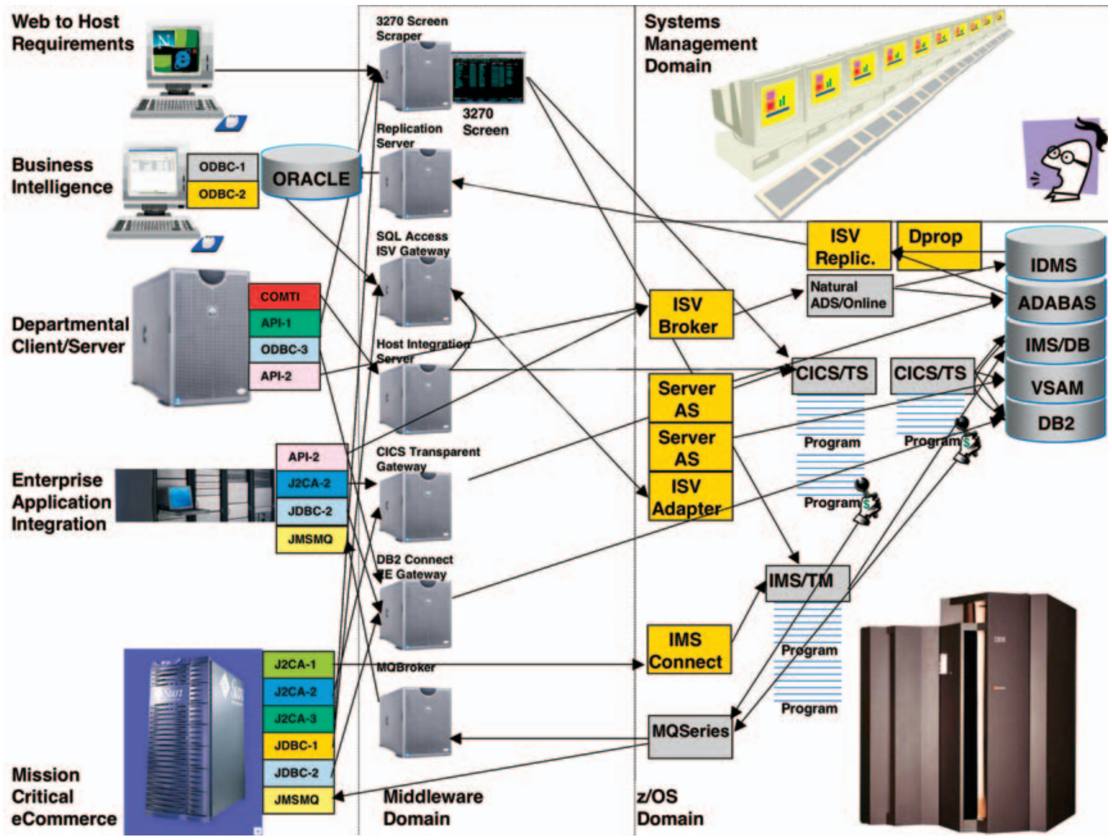

Figure 1: Common mainframe integration implementation

The diagram is by no means as ridiculous as it may look. Most large organizations, which have been asked to support individual project requirements for mainframe integration, will have a set of technologies deployed at least as varied and incompatible as the diagram suggests. Such enterprise architecture puts tremendous hard and soft cost pressure on IT and is unlikely to provide the service that the application teams expect.

Mainframe Integration

Part of the challenge in approaching mainframe integration, as a discipline all of its own, is to define what is meant by mainframe integration. As has been stated already, most projects will look for integration with DB2 or Adabas, or VSAM or CICS as separate disciplines, when in reality they are all elements of the higher-level discipline of mainframe integration.

Taxonomy of Mainframe Integration

This section defines the higher-level requirements that should be set forth in any charter for mainframe integration. It uses a well-established categorization in order to make the options more accessible.

Mission-Critical Integration

Mainframe integration needs can be considered mission critical if the applications they support have high-availability, high-volume, transactional requirements – where a service interruption or data integrity issue costs money in real time.

Informational Integration

Mainframe integration needs can be considered informational if the applications they support are generally read-only in nature and exist to support internal or planning-type applications – where a service interruption should not impact revenue in real time in any way.

Mainframe Integration: Taxonomy

Typical commercial uses:	• Data replication • J2EE applications • Business activity monitoring • Business process management	• Business intelligence • .NET applications • Integration Servers/Message brokers • Event-driven systems

Standard APIs

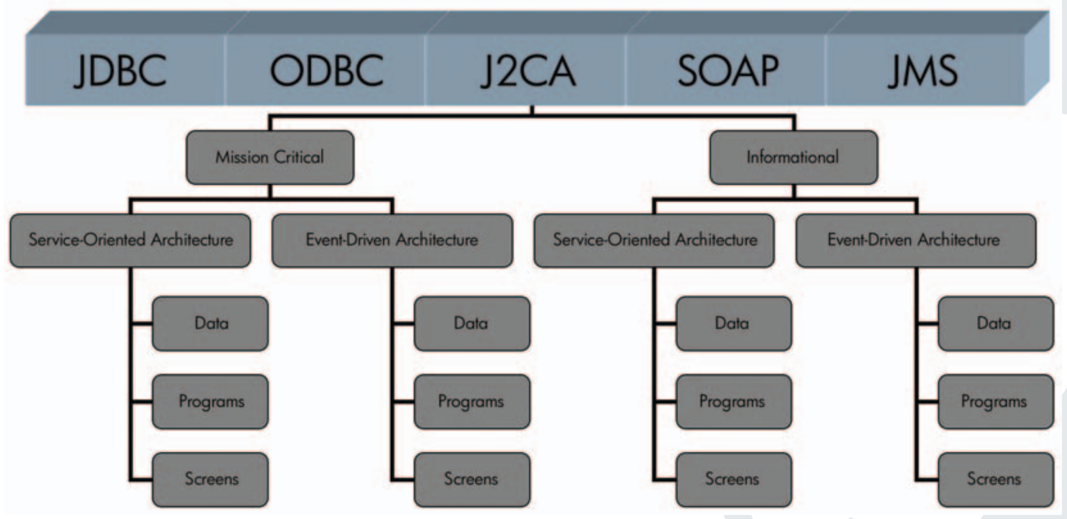
JDBC
ODBC
J2CA
SOAP
JMS
Mission Critical
Informational
Service-Oriented Architecture
Event-Driven Architecture
Service-Oriented Architecture
Event-Driven Architecture
Data
Programs
Screens

Figure 2: Accepted categorization for mainframe integration

Service-Oriented Architecture

Service-oriented architecture is the accepted industry model for representing applications on the network as reusable services. Various interface standards exist for these services. Such standards range from DCE/RPC and CORBA, through database stored procedures to more recently Web services. In all cases, requests are made of the services by a calling application that expects some form of reply; consequently, such architectures are often referred to as Request/Reply. A fundamental element of a service-oriented implementation is the separation of interface from implementation. This enables a looser coupling between the service consumer and the service provider. The separation is enabled by means of a metadata layer that makes the interface definition available at design time to development professionals building consumer applications.

Event-Driven Architecture

Event-driven architecture is designed to support a more real-time method of integrating application processes. Events occur throughout existing applications. Events are largely defined by their meaning to the business and granularity. For example, for some a database update may be a meaningful event; for others the successful completion of an order process may be an event. Regardless of the granularity of the event, event-driven architecture concerns itself with ensuring that interested parties, usually other applications, are notified immediately when the event completes. This is in contrast to traditional methods of communicating such events, which rely on "batching" events and communicating them periodically. Other approaches have involved repeated Request/Reply activity against certain database artifacts to detect changes. A successful event-driven architecture can identify granular events, and enable administrators to determine what level of event aggregation or context is required to convey business meaning. It must also be driven using a "push" method of communication rather than request/reply. An event-driven architecture must also be able to publish events to multiple interested parties.

Requirements

The following section identifies the most common specific requirements for integrating distributed computing environments with applications and data resident on mainframes. Taken as a whole, it can be viewed as the strategic set of needs for a mainframe integration solution. For many organizations, however, each individual requirement is treated as an integration requirement of its own. As pointed out before, such an approach leads to a highly complex, fragmented and unresponsive solution to the broad requirements for mainframe integration.

Service-Oriented Architecture

Web-to-Host

Web-to-Host is one of the most common, first-stage mainframe integration initiatives undertaken. It benefits from being totally non-invasive insofar as there is zero impact on the existing mainframe systems. The objective of

Web-to-Host integration is merely to replace the black/green 3270 image of a mainframe application with a Web browser rendition of the same screen. The benefits are presumed to include a more consistent look-and-feel across all the users' desktop applications as a result of the elimination of 3270 emulation software. There is no change anticipated in the usage profile of the mainframe application since a single 3270 interaction is replaced with a single browser-based interaction.

Aggregated Access to CICS, IMS/TM or IDMS Screens

Using a similar technology to Web-to-Host, aggregated access to mainframe screens allows a screen navigation scenario to be encapsulated within a single browser interaction. This is achieved by having a 3270 scripting function navigate menus and fields in response to a buffer of information sent from the browser. The benefits of such process reengineering include a more streamlined and less keyboard intensive experience for the user. Sophisticated examples of such aggregation also include navigation from other mainframe systems. This results in the delivery of entirely new business applications that fuse disparate applications "at the glass."

SQL Access to DB2 Data

SQL access is a very common requirement where the relational structures, within the dominant mainframe relational database of DB2, are exposed directly to Business Intelligence and application platform environments. It is most commonly implemented for read-only purposes; however, an increasing number of implementations are emerging that require update access. Consequently, the issue of data-integrity is often raised. From a technological perspective the means support for mainframe transactions, controlled by z/OS Resource Recovery Services (RRS) and initiated from XA compliant, two-phase commit capable transaction coordinators.

SQL Access to Non-Relational Data

In lieu of replicating non-relational databases to local relational or mainframe resident DB2, an increasing number of organizations are looking to provide direct relational access to non-relational mainframe databases. The most common of which include IMS/DB, Adabas, IDMS and VSAM. In order to achieve SQL access to non-relational data, the non-relational structures need to be mapped into a relational form, and metadata describing such mapping exposed to the composite application. Again, in most cases this is read only due to the tight coupling of non-relational implementations and the mainframe-resident applications that drive them. However, there are a significant minority of needs that embrace update activity. Such activity demands the same transactional support identified above.

Transactional Access to CICS or IMS/TM Transactions

Perhaps the most common form of mainframe integration is new, mission-critical eCommerce applications within application or integration servers, which need to reuse business rules embedded with existing CICS or

IMS/TM applications. These systems usually have high-volume, high-availability needs and are almost always transactional in nature.

Web Services Access to Mainframe Applications

Web services hold out the promise of a more flexible way for applications to interoperate. With so much invested in mainframe applications it is hardly surprising that Web services access is being considered by many organizations as a panacea for legacy reuse. The Web services integration metaphor allows complex mainframe interface implementations to be wrapped in a standard manner, which is compatible with most non-mainframe development tooling.

Reuse of Legacy 4GL Applications

Many mainframe business applications still run within the popular 4GL environments of Natural, ADS/Online, IDEAL and Mantis. These systems were often bundled in with non-relational database systems popular during the 1980s. For some organizations interaction with such systems is an important way of leveraging legacy value.

Mainframe Access to Distributed Assets

With an increasing proliferation of applications and data resident on non-mainframe environments, it is not surprising that mainframe applications could benefit from access to such systems. Often referred to as "call-out," such requirements include mainframe applications synchronously calling applications resident in J2EE or .Net application servers or making SQL requests to remote Oracle or SQL/Server systems.

Event-Driven Architecture

Asynchronous Access to CICS or IMS/TM Applications

Asynchronous communication has emerged from its less-than-auspicious beginnings as a successful and important mechanism for loosely coupling applications that were never designed to interoperate in a synchronous manner. Where mainframe systems are concerned, asynchronous access is most commonly implemented in support of Business Process Management (BPM) environments that implement workflows, which imitate human interaction with various systems. In order to deliver a similar transactional capability that a synchronously bound human operator can provide, the common network transport for such interaction is WebSphere/MQ. This ensures the BPM technology can handle positive and negative acknowledgments, for asynchronously initiated mainframe applications, in a timely and reliable manner.

Initiation of Mainframe Batch Processes

Certain batch processes are dependent upon the arrival of information from other platforms. As integration needs shift in support of a more real-time business environment, the need to initiate elements of the batch process ahead of the overnight cycle becomes commonplace. In conjunction with proliferation of BPM tools it is important to enable off-mainframe event drive systems to trigger batch processes in the same "on-demand" way online applications can be initiated (see above).

Data Synchronization

Despite the utopian objective of having only a single live copy of any data artifact, it is often desirable to take copies of the core systems of record for Business Intelligence purposes or simply in support of a new application for which practical, direct integration with the mainframe is impossible. In such situations it is critical for the accurate operation of the system, dependent upon the replicated copy of data, that the data itself is kept synchronized with the system of record.

Event Triggering for Business Process Management (BPM) Systems

Perhaps the highest value form of event-driven architecture for mainframe integration is the ability to trigger new-business processes, developed with BPM tools, based on events occurring within mainframe systems. Mainframes still run the majority of commercial applications; consequently, considerable value can be unlocked from the operation of such applications if the events they process are exposed to BPM environments. Fairly significant technical barriers exist to achieving such integration in a flexible manner. Few mainframe applications were designed with such integration in mind; consequently, such triggering requires code-modification in most cases. With availability of source code uncertain for legacy applications, such modification is troublesome and often impossible.

Common Approaches

Summarized here are the more common integration approaches to the tasks identified above. None of the approaches identified below can be considered a solution to the overarching challenge of mainframe integration. Most solutions are deployed in support of a narrower integration need such as 3270 screen scraping, programmatic integration with CICS applications of SQL access to mainframe DB2 databases. On completion of the section it is worth reflecting on just how many technologies and vendors are required to satisfy the broader needs of mainframe integration.

Service-Oriented Architecture

Web-to-Host

Rudimentary screen-scraping capability can be achieved with many terminal emulation products; however, for reasons of deployment efficiency, the function is more commonly achieved via a mid-tier technology running within a Web server dedicated to the Web-to-Host function. Such implementations also demand a dedicated server environment with at least one hot-standby environment.

Aggregated Access to CICS, IMS/TM or IDMS Screens

Implemented in a similar way to the Web-to-Host systems, screen aggregation solutions will have a separate server footprint and web server environment. Resource demands are invariably higher than Web-to-Host since simulated screen navigation, for applications with high-user concurrency, can be resource intensive. Most production implementations will be running on clustered high-specification server systems. Such integration technologies often provide a programmatically callable API (commonly J2CA) to allow new applications to drive the aggregated screen interactions.

SQL Access to DB2 Data

IBM's DB2Connect product is the most widely implemented technology to support SQL access to DB2. Low-volume, read-only requirements are satisfied by DB2Connect personal edition; however, once a transaction need is established, organizations have to implement the server-based DB2Connect Enterprise Edition. This requires the implementation of a high-performance server machine, with hot-standby or clustered machines for failover and scalability requirements. In addition, for SQL access via this Enterprise Edition gateway server, an ODBC or JDBC driver needs to be installed and configured on the calling application platform.

SQL Access to Non-Relational Data

Two fundamental methods exist to satisfy this requirement. First, the data is periodically replicated to either a mainframe-resident DB2 database, or off-mainframe to a distributed Oracle or SQL/Server database. This requires the installation of some Extract, Transformation and Load (ETL) type technology that can access both the non-relational database and the relational database, wherever it may reside. It also requires the installation of JDBC or ODBC drivers to access the newly created, relational copy of the non-relational data. Second, technology exists to query the non-relational data in place by translating SQL calls into the native calling syntax of the non-relational database. Such solutions often involve the deployment of a server machine that intercepts SQL calls via a specific ODBC or JDBC driver, and then passes an internal form of the request to a mainframe resident series of address spaces to satisfy the SQL call.

Transactional Access to CICS or IMS/TM Transactions

IBM and a range of other vendors, including Microsoft, provide solutions for this common requirement. Once again a gateway server, with attendant clustering and failover, is required to translate requests from the calling application platform into a form that can be handled by the CICS or IMS/TM application. Such products are often sold at the departmental level, and as a result it is common for an organization to have several flavors of such a solution simultaneously active.

Web Services Access to Mainframe Applications

A relatively new integration approach, most current implementations are extensions of screen-scraper products. However, direct Web services access to the mainframe (without the cost and complexity of yet another gateway machine) seems to be what the industry wants. IBM has its CICS/SOAP toolkit which requires a commitment to transaction modification. In addition there are several ISVs that have released software that runs either natively on the mainframe or inside CICS. This software can map SOAP requests dynamically to the format required by the legacy transactions.

Reuse of Legacy 4GL Applications

Most vendors of 4GLs offer some form of mainframe integration technology designed specifically to integrate with the 4GL programs. Examples include EntireX Broker from Software AG. Such products exploit interfaces not normally visible to technologies that integrate with CICS in the usual manner, as the 4GL interfaces are often hidden from standard CICS interfaces. The popularity of these languages, and their enduring value within large organizations, makes re-use attractive. Consequently, it is likely that one of these mainframe integration technologies will find its way into the software portfolio of companies with significant mainframe 4GL legacy systems.

Mainframe Access to Distributed Assets

Most current implementations of mainframe access to distributed applications and data are "homegrown." Often such implementations involve modification to mainframe programs to call local network services. A complimentary "catcher" program is also required on the remote platform. Recently some vendors have begun implementing Web services-oriented access from CICS to remote Web services using code "stubs" inside CICS that generate SOAP calls to remote platforms automatically.

Event-Driven Architecture

Asynchronous Access to CICS or IMS/TM Applications

The WebSphere/MQ bridge for both CICS and IMS/TM provides a convenient way for mainframe applications to be the receiving point for an event-driven architecture. Messages received by the bridge are mapped to CICS and IMS message-oriented applications. In practice the mapping process is largely manual. Facilities exist within the bridge to allow an organization to code its own mapping routines. Such code needs to be sensitive to the location of fields that require data-type conversion and must itself handle that conversion. Exploitation of the bridge is therefore not a task to be undertaken lightly.

Initiation of Mainframe Batch Processes

Distributed job scheduling products can intercept requests from remote systems and initiate mainframe batch processing suites. These products typically reside on middle-tier servers and communicate with mainframe resident counterparts. A more common approach is to use WebSphere/MQ to trigger batch jobs. This is a more manual and development-intensive process.

Data Synchronization

Over the years various data replication technologies, such as IBM Data Propagator, have been employed to move copies of mainframe data onto distributed platforms to make access to the data more efficient. ISVs have also participated with products from companies such as Sybase and Striva. The biggest headache with all implementations has been the timeliness of keeping the copies synchronized. In virtually all scenarios, changes to the source data are accumulated, batched and loaded into the remote system on a periodic basis. As business demands a more real-time enterprise, with zero-latency for corporate information, this model is tending toward obsolescence.

Event Triggering for Business Process Management (BPM) Systems

The most common way for mainframe applications to trigger processes defined within BPM tools is to modify the application code to put messages onto a WebSphere/MQ queue. This is a perfectly functional solution but requires a commitment to the ongoing support and maintenance of the modification. Furthermore, it does not provide a flexible platform for event publishing. If the published message needs to be enriched, further code modification and testing is required. Finally, as event publishing to BPM tools demonstrates its true value, the business groups will demand more complex event functionality. This will place an enormous ongoing development burden on those responsible for modifying the legacy applications.

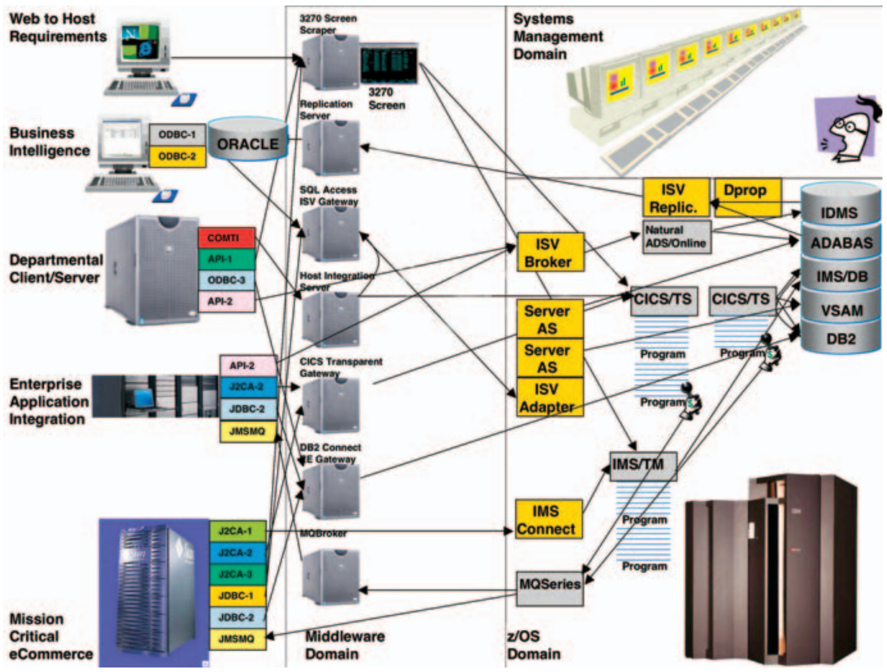

Figure 3: Reprise of typical mainframe integration scenario diagram

Summary

Against the backdrop of the preceding discussion, the diagram (above) becomes more meaningful. When the full range of technologies deployed to support any interaction, which could be characterized as mainframe integration, is assessed, the implementation complexity for most companies is staggering.

Composite Applications

While, from the perspective of the mainframe systems administrator, the above catalog of solutions to the mainframe integration challenge may be functionally satisfactory, it is worth looking at how the application developer of the new composite applications is likely to exploit such an architecture.

Application Development Perspective

In most situations when application development or systems integration professionals working within a J2EE or .NET application platform environment need access to applications or data on remote platforms, those platforms manifest themselves to the application platform through a single interface, commonly referred to as an adapter. The more types and implementations of an adapter, the more complex it will be for an application platform developer to build and test new applications. The diagram in Figure 3 shows that treating mainframe integration as a collection of unrelated challenges is almost guaranteed to force an application platform developer to confront a range of differently implemented Adapters to satisfy access to a single mainframe environment.

Metadata

For developers, who work within sophisticated integrated development environments (IDEs), the ability to access metadata in a consistent manner is fundamental to the smooth exploitation of remote data sources. Using a collection of methods to integrate with mainframes will expose metadata in an inconsistent manner. Furthermore, many mainframe integration vendors offer specific extensions to common IDEs to make manipulation of metadata easier. If a developer is faced with a different IDE extension for each z/OS datasource type, metadata management becomes more of a chore.

Transactions

A crucial point for any business-critical application is the management of data integrity. For composite applications, data integrity is managed via synchronous two-phase commit (2PC) transactions. 2PC is a complicated mechanism that manages data integrity across multiple systems by relying on support for transactions in the underlying system. Such transactional support will routinely maintain locks for the duration of the transaction as it traverses all the platforms within the transaction boundaries. For two separate resources this is generally acceptable; however, as the number of separate resources involved in the transaction grows beyond three, lock durations grow geometrically. Such extended periods of locking can have profound effects on concurrency. As we have seen, traditional mainframe integration approaches treat different mainframe systems as separate integration challenges. This has the potential to create a transactional model with many separate resources to be coordinated on a single mainframe image. Using 2PC to coordinate transactions across multiple mainframe resources, from the perspective of the application platform, is likely to create significant lock duration issues. From the perspective of the mainframe this may create an untenable concurrency issue as many of the traditional workloads will be impacted.

Fortunately IBM has introduced a capability within the z/OS operating system known as Resource Recovery Services (RRS) that can coordinate updates across many mainframe systems using only a single commit phase from the perspective of the composite application. This dramatically improves concurrency; however, it can only be achieved if mainframe integration is approached as a single challenge technologically rather than a collection of unrelated disciplines. There are important technological prerequisites that the mainframe integration software must possess in order to allow the entire mainframe to be treated as a single resource manager. It is a technical impossibility for such functionality to be available if mainframe integration is spread across several disparate mainframe integration tools.

Systems Administration Perspective

End-to-end visibility throughout the entire application life cycle is of well-understood value in the J2EE and .NET development world. Many integrated tools exist that allow programmers to rapidly develop, deploy, test and manage composite applications. From the mainframe perspective an equally bountiful range of tools can be found that display activity against mainframe systems down to a fundamental level. For the most part these two worlds do not intersect in any meaningful way from a systems management perspective. Consequently, when the newly developed J2EE or .NET transaction leaves the application platform on its way to the mainframe, the developer or systems administrator must switch systems management tools to ensure its activity can be tracked and, in the event of a failure, diagnosed. Although most mainframe integration technologies offer some form of additional systems management capability to plug the gap between the transaction leaving the application platform and being executed within the target mainframe subsystem, there is little default integration between these tools and the various juxtaposed systems management layers. Consequently, it is impossible for the J2EE or .NET administrator to manage the end-to-end process, since the number of tools to learn is bewildering.

Occam's Razor Applied to Mainframe Integration

Faced with common implementations of mainframe integration, William of Occam would ask several key questions:

- Can the same function be achieved with fewer moving parts?
- Are so many middleware products required?
- Are so many vendors required?

Occam's Ideal Solution

A Single Product to Support all z/OS Subsystems

One solution is a single middleware product that provides integration with CICS/TS (programs and screens), IMS/TM (programs and screens), IDMS/DC (programs and screens), Natural programs, standard batch programs, DB2, IMS/DB, VSAM, IDMS/DB, and Adabas, etc. Essentially, this is a single product that addresses all mainframe data, program and screen interface requirements.

A Single Product to Implement an Event-Driven Architecture and Service-Oriented Architecture

This solution supports service-oriented architecture APIs such as J2CA, JDBC, ODBC and Web services. It also supports event-driven architecture by enabling mainframe events and data changes to be captured non-invasively and published to an Enterprise Service Bus such as WebSphere/MQ for consumption by a wide range of technologies such as Business Process Management, Business Activity Monitoring or Enterprise Application Integration. Support for both event-driven architectures and service-oriented architectures should be described internally by means of standards based metadata for maximum interoperability with existing integrated development environments.

A Single Product to Handle Exposed Transactional Controls to Distributed Applications

The solution supports both synchronous, two-phase commit and asynchronous transactions within one product. The product should have the ability to allow updates to any combination of z/OS resources, within a single operating image, to be transactionally coordinated using a single-phase transaction by the distributed platform. Asynchronous transaction support is required to ensure the integrity of an event-driven architecture.

A Single Consistent Security Implementation

The idiosyncrasies of each mainframe subsystem security implementation should manifest themselves in a consistent manner to the distributed application platforms. In addition, a simple and consistent method of exploiting non-mainframe authentication and auditing inside the mainframe should be provided. This will ensure J2EE and .NET-based applications do not introduce vulnerabilities as a result of having to learn several different ways of bridging the security gap of the mainframe and distributed world.

A Single Integrated Approach to End-to-End Systems Management

With a single product implementation for mainframe integration should come a single interface to manage the transactions from the point they touch the integration software to the point they are returned to the calling application. In addition, an architectural awareness needs to enable systems management tools, which monitor the J2EE and .NET application platform, to query the instrumentation inside the mainframe integration software. This provides the systems administrator with the opportunity to visualize transactions from an end-to-end perspective within a single tool.

A Pricing Model That Enables Project-Oriented Purchases

While a single, all-embracing product for mainframe integration offers significant benefits to any organization with considerable mainframe workload, it cannot be burdened with a licensing model that prevents small projects, with tight ROI constraints, from purchasing it.

Shadow – The Only Integrated Solution for All Mainframe Integration Requirements

Shadow is the only integrated technology to deliver support for the entire range of mainframe integration needs. With hundreds of the world's largest organizations running mission-critical applications, supporting in many cases thousands of transactions per second, Shadow's breadth of functionality is matched only by its suitability for the most demanding of mainframe integration scenarios.

Within a single technology, Shadow enables any distributed J2EE or .NET application to exploit mainframe screens, programs and data in a consistent manner using J2CA, JDBC, ODBC or Web services. Additionally Shadow enables the non-invasive capture of mainframe events from within any supported environment, with publishing of XML representations of those events onto an Enterprise Service Bus such as WebSphere/MQ. The entire environment is transactionally controlled to ensure data integrity. In addition Shadow has a consistent and robust security model which blends the best mainframe security with the demands on distributed systems, to provide a vulnerability-free integrated environment. Finally, the entire product is managed through a single management console that presents an end-to-end view of transactions running between the mainframe and the application platform. The instrumentation gathered by Shadow for systems management is available and visible in popular J2EE systems management tools such as Wily Enterprise.

Importantly, Shadow can be licensed on a project-by-project or application-by-application basis. This ensures that organizations can benefit from a single mainframe integration architecture without the barrier of huge software licensing costs to contend with.

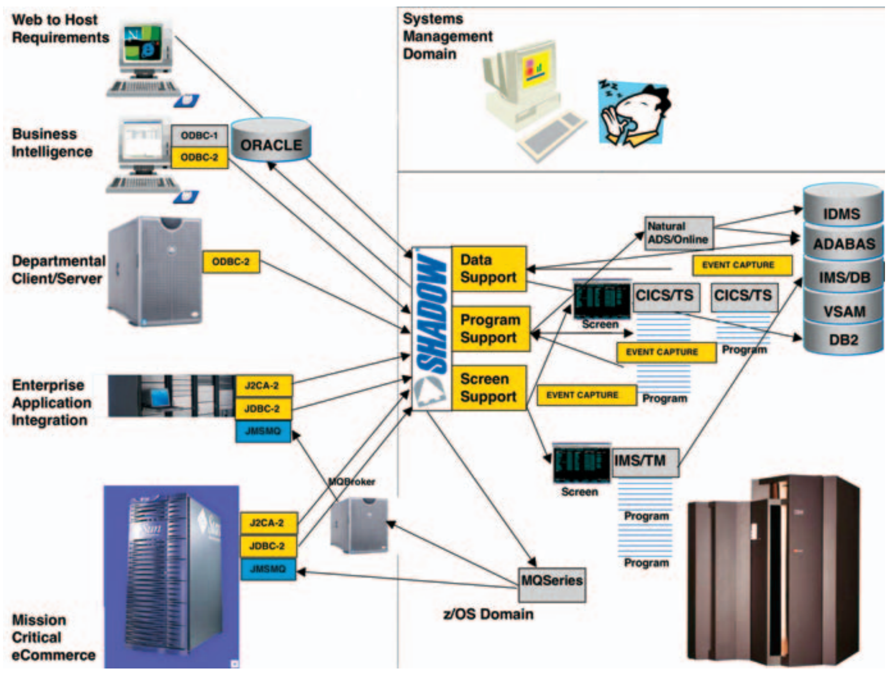

Figure 4: Shadow-integrated approach to mainframe integration

The Benefits of Simplicity

By implementing Shadow, organizations can begin to accrue the many benefits associated with a simpler approach to mainframe integration. Ultimately these benefits fall into the categories of the cost of complexity outlined in an earlier section. For consistency the benefits are organized below along the same lines.

Hard Costs

Software

Although Shadow itself carries a software license fee, it pales in comparison to the fully burdened software license costs of the combination of software required to provide the same range of functionality. Looked at from the perspective of ongoing software maintenance charges, Shadow can often prove to be a more cost-effective solution than the collection of incumbent mainframe integration products. Furthermore, Shadow can be licensed in a piecemeal fashion.

Organizations can acquire Shadow on a project-by-project basis, yet ultimately own a single unified architecture for all mainframe integration needs.

Hardware

Shadow requires no additional hardware to function. Essentially, all the costs associated with the deployment of hardware gateways can be avoided. This represents a considerable hard-dollar cost savings for organizations with even a moderately diverse mainframe integration requirement. If the integration requirement has a mission-critical dimension, the savings begin to grow exponentially as fail-over and redundancy gateways can be avoided.

Soft Costs

Business Agility

A simpler, proven architecture can make the deployment of new business applications, which have mainframe dependencies, a non-issue. This allows project teams more time to focus on the real issue of building high-value systems to support new business needs. Having to learn a new integration tool for each new requirement introduces risk and friction into composite application projects. This translates directly into costs to the business as systems take longer to implement and may have substandard systems management capability or exploitation.

Personnel

A single architecture for mainframe integration will require considerably less configuration and administration personnel. Many organizations maintain ratios of management staff to hardware and software components. Eliminating unnecessary hardware and software platforms will, if these ratios are honored, translate directly into savings. In addition less personnel time will be spent deploying the application for reasons alluded to above.

Availability

A simpler architecture should be far more reliable, particularly if the systems management tooling can be used to diagnose the environment in a holistic manner. Improved availability for mission-critical applications translates directly into improved operational performance, which, depending on the application, can have a profound impact on an organization's revenue-generating potential and general customer satisfaction.

Summary

Mainframe integration demands a strategic approach if it is not to become the weakest link in end-to-end integrated systems, which are heavily dependent upon mainframe assets. However, the unique workload mix of the mainframe, and the expectations of non-mainframe application developers and systems integrators, push mainframe integration architectures inexorably toward complex implementations. When the range of integration needs is considered in a holistic fashion, rather than as a group of unrelated application/subsystem pairs, it can be seen that a practical, cost-effective and agile architecture is possible for what is certainly the most important integration task for almost all large organizations and government agencies.

The good news is that Shadow, from NEON Systems, has been developed to provide a solution to the entire range of mainframe integration needs from within a single product architecture. With diagnostic integration to popular application platform systems management tools, Shadow dramatically improves service availability across the entire range of composite applications that depend on mainframe assets. The cost savings from a strategic approach to such a complex integration task can be huge. Furthermore, such an implementation would certainly be considered consistent with Occam's Razor, and thus the greatest scientific minds of our era; for certainly using Shadow it is possible "*to do with fewer that which is done in vain with more*" (sic).