IDC OPINION

Enterprise applications are designed for finite life spans, but these life spans can be unexpectedly long, potentially in excess of 30 years in the case of some mainframe custom-developed applications. While such applications can have architectural challenges and require continual care and feeding by application development teams, the code bases are often extraordinarily mature and bug free, and the degree to which the applications fit their requirements and accurately embody enterprise business rules is unrivaled. IDC finds that in many cases, enterprises can take steps to modernize these applications, often referred to as “legacy applications,” without unnecessary disturbance to the code base and often in ways that permit aggressive evolution to suit changing commercial or technological needs. In particular, IDC recommends the following in dealing with legacy applications:

- Enterprises should engage in a process of surveying and analysis in managing internal application portfolios, classifying and grouping them according to various organizational schemes intended to identify the best approaches for evolving or modernizing them. Such analysis is key to prioritization and cost containment of modernization projects.

- For a certain class of legacy applications found to meet certain fitness criteria, enterprises should consider shifting the application development process to more cost-effective and agile hardware and software application platforms. Such an approach can reduce the cost of maintenance, improve development efficiency, and even widen the available resource pool.

- For applications requiring a significant degree of evolution and modernization, enterprises will find off-mainframe application testing and quality tool chains more vigorously available in the market, more aggressively supported by vendors, and more productive to use and manage over the remaining lifetime of the applications in question.

- For a certain class of legacy applications found to meet certain fitness criteria, enterprises should consider approaches that provide a choice for application workload deployment across cost-effective hardware and software platforms. The benefits of such transitions in where an application is deployed can range from reduction in mainframe capacity needed to allow the mainframe to focus on critical core system execution to genuine flexibility of workload deployment to achieve higher levels of cost efficiency and commercial agility.
# TABLE OF CONTENTS

## In This White Paper

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Situation Overview</td>
</tr>
<tr>
<td>2</td>
<td>Understanding Code Assets</td>
</tr>
<tr>
<td>3</td>
<td>Application Atrophy and the Need for Modernization</td>
</tr>
<tr>
<td>3</td>
<td>Symptoms of Application Atrophy</td>
</tr>
<tr>
<td>3</td>
<td>Micro Focus Enterprise Analyzer</td>
</tr>
<tr>
<td>4</td>
<td>Development Kickdown</td>
</tr>
<tr>
<td>4</td>
<td>Benefits of Off-Mainframe Development</td>
</tr>
<tr>
<td>5</td>
<td>Micro Focus Enterprise Developer</td>
</tr>
<tr>
<td>5</td>
<td>Testing Unplugged</td>
</tr>
<tr>
<td>5</td>
<td>Benefits of Off-Mainframe Testing</td>
</tr>
<tr>
<td>6</td>
<td>Micro Focus Enterprise Test Server</td>
</tr>
<tr>
<td>6</td>
<td>Deployment Downsizing</td>
</tr>
<tr>
<td>6</td>
<td>Harnessing Modern System Cost Curves</td>
</tr>
<tr>
<td>7</td>
<td>New Application Deployment Options</td>
</tr>
<tr>
<td>8</td>
<td>Deployment Agility</td>
</tr>
<tr>
<td>8</td>
<td>Micro Focus Enterprise Server</td>
</tr>
<tr>
<td>8</td>
<td>Future Outlook</td>
</tr>
<tr>
<td>8</td>
<td>Preparing for the Clouds</td>
</tr>
<tr>
<td>9</td>
<td>The Coming Mobile Client Platform Revolution</td>
</tr>
<tr>
<td>9</td>
<td>Challenges/Opportunities</td>
</tr>
<tr>
<td>9</td>
<td>The Shrinking COBOL Developer Pool</td>
</tr>
<tr>
<td>9</td>
<td>Architectural Dead Ends</td>
</tr>
<tr>
<td>10</td>
<td>Conclusion</td>
</tr>
<tr>
<td>10</td>
<td>Learn More</td>
</tr>
<tr>
<td>10</td>
<td>Related Research</td>
</tr>
</tbody>
</table>
IN THIS WHITE PAPER

This white paper examines key approaches to application modernization and evolution for certain IBM mainframe applications with long life spans. It discusses the challenges of legacy applications and the available modernization approaches to handle these challenges. The paper also analyzes in some depth specific approaches that keep application logic largely intact or allow it to evolve incrementally and productively.

SITUATION OVERVIEW

Information technology (IT) is in a constant state of flux. Change is considered progress, and the velocity of change appears to be constantly accelerating. While there are incontrovertible advantages of a new generation of technology compared with even its immediate predecessor, the cost of simultaneously managing multiple generations of technology, managing the transitions between them, and managing the changes in the layers of technologies that host them, such as in the underlying application platforms and hardware, can outweigh the productivity or cost advantages of new technology adoption. An example of an architectural transition whose advantages are considerably lower than initially predicted is the jump to two-tier client/server models that took place throughout the early to mid-1990s. In recent years, many of the same companies that deployed such technologies have invested in transitions of various types that often lead to the replacement of such applications. The constant adoption of new technology has resulted in a severe case of enterprise technological sprawl. Additionally, this condition looks set to worsen (Web, mobile, social media, and big data together are causing unprecedented levels of change and consumer demand). While it is essential to experiment and inevitable to have a diverse mix of technologies, the sprawl has not resolved the issue of whether one type of technology is contextually superior to another. Enterprises are wise to approach the issue of modernization in a level-headed manner and work through their application portfolios using a methodical process of analysis.

Understanding Code Assets

Enterprises live and die by their application portfolios and have come to understand that their code assets are among the most valuable assets they have. It stands to reason then that enterprises expend considerable resources in maintaining the viability of their application code in the face of rapidly changing circumstances. Application portfolio modernization is a complex, continual exercise that enduring enterprises engage in, bringing a variety of approaches to the task. Some of the typical modernization options facing enterprises are as follows:

- Map enterprise requirements and workflows to existing applications and tier applications along various metrics of scale, complexity, mission criticality, or interoperability needs

- Analyze and group applications into potentially overlapping sets of functions that might have emerged independently in separate divisions or through mergers and acquisitions
Assess duplicative applications to understand their functional coverage and reach in order to determine which might be evolved to usurp others

Uncover unmet application needs and engage in a process of assessment to determine if new applications should be developed or existing applications should be evolved

Categorize existing applications into different groups with different modernization approaches (These modernization approaches can run the gamut from planned obsolescence and scheduled replacement to a variety of service-oriented encapsulation techniques to approaches where the code is essentially preserved and evolved.)

Application Atrophy and the Need for Modernization

The degree to which an application fits its requirements does not remain constant throughout the application's life span. Over time, applications drift into a state of atrophy that necessitates modernization. The following types of changes in the IT environment can cumulatively lead to a buildup in application atrophy:

- **Changes in user interaction (UI) styles.** Changes in UI style, whether through transitions in client platforms or shifts to more portable UI frameworks such as Web browsers, are continual and can lead to episodic transformations such as from command line– or function key–oriented systems to graphically oriented systems. Today we are experiencing such a transformation as applications shift to touch or gesture modalities of interaction and their delivery gravitates toward mobile devices. Application UI models are stratifying along major successful device ecosystems (e.g., Apple, Android, Windows) and a somewhat lower functionality model based on industry-standard HTML5/CSS/JavaScript, which is suited for less device-specific use.

- **Changes in hardware architectures.** The fact that hardware has a more finite life span than software and requires physical upgradability to harness improvements in performance can lead to significant stress on applications. As hardware moves through its rapid changes, operating systems that are closely tied to hardware architectures can undergo significant transitions or can suffer from inadequate vendor investment. One only has to look at the varied integrated systems that have reigned during various eras, such as Pick, DEC VAX, or HP3000, to appreciate the stresses that hardware and operating systems place on the application.

- **Changes in software architectures.** Changes in structure and connectivity of computing systems brought about by evolving networking characteristics, including bandwidth, speed, latency, and reliability, create opportunities for change and new capabilities that are often unimaginable at application inception. Distributed systems were highly suspect architectures 30 years ago because IT practitioners of the day did not have access to and could not imagine networks with the ubiquity, throughput, latency, and reliability of today's Internet. Today, software architectures that utilize massive portfolios of cloud services and data and connect them with diverse mobile devices with location intelligence and offline capability are essentially revolutionizing what is possible with enterprise applications.
Changes in programming languages and programming models. While programming languages shift over longer time frames, measured in multiple decades, they also place stresses on applications by gating the supply of developers who can maintain them. Developer ecosystems shift slowly, governed by their ability to learn and transition between different programming languages and the attendant skills they require (e.g., object orientation). As these ecosystems shift, they generate low-level stresses on application maintenance projects that are manifested by the ability to find and retain developers. COBOL is an example of a technology that is under relative stress today because it is rarely taught in schools anymore. While the situation is helped by improved human longevity and increasingly delayed retirement of professionals due to economic circumstances, solutions must be found to attract skills to COBOL if a significant body of applications is to be maintained.

Symptoms of Application Atrophy
Understanding the processes by which applications can degenerate in these ways can lead to a better understanding of the need for modernization and detection of the more direct symptoms that characterize the need for modernization. A modernization process of some type is often the remedy when one or more of the following problems are encountered with a particular system or body of code:

- Existing maintenance costs that are unsustainably high
- Rigid systems that are resistant to change or adaptation to new requirements
- Isolation or lack of interoperability that necessitates an investment in considerable additional glue to connect with other enterprise systems

An application modernization approach is often recommended when any of these three critical symptoms are observed in the analysis process. The discovery and assessment process that enterprises engage in for enterprise applications also measures the degree of change needed for each application, which becomes an essential input into which approach to take in modernizing.

Micro Focus Enterprise Analyzer
Micro Focus Enterprise Analyzer is one of the leading tools for application portfolio analysis in the industry. It enables the acquisition of the knowledge needed to successfully conduct a modernization analysis. Its key capabilities include the identification of development priorities and the generation of portability assessments and modernization candidates. Additional tools from Micro Focus help with application code analysis, which often is part of the modernization assessment. The Micro Focus analysis tools are designed to help in aligning business priorities, documenting application behavior, assessing major system change impact, and improving code quality practices. Overall, these tools support the application modernization process by providing a mechanism for unlocking the complexity, containing maintenance costs, and prioritizing projects.
Development Kickdown

A portfolio assessment involving legacy hardware running older applications can yield different outcomes for different parts of the application portfolio. Some outcomes may reflect operational challenges around current application development processes. Such challenges may include poor throughput of system updates, low quality ratings, inadequate rates of change, or high cost per revenue ratings for application development. One potential modernization opportunity that can yield surprising productivity savings is the relocation of the development process to modern platforms.

The mainframe is a finely tuned machine for running production workloads. It excels at reliability, security, and the processing of large-scale, high-volume transactions. Mainframe application development and unit testing processes can tie up significant mainframe capacity, and mainframe processes require mainframe skills over and above those needed to support the programming language of the code itself. Such skills are increasingly harder to come by because mainframes are largely unfamiliar to recent generations of programmers, and the skills are rarely, if ever, taught in computing education courses. Skills issues conspire to make mainframe application maintenance unnecessarily costly and have created an opportunity for tool vendors to devise interesting solutions for conducting the development process off of the mainframe.

Benefits of Off-Mainframe Development

Performing mainframe application development away from the mainframe environment can yield a variety of advantages, including:

- **Cost savings.** Depending on the size of the development team, removing application compilation and unit testing workloads can tangibly reduce the number of mainframe MIPS required to run the mainframe. This can lead to cost savings not only in hardware but also in software licensing because most mainframe software is still licensed on the basis of mainframe capacity.

- **Developer productivity.** Modern developers are trained to work with tools that live natively on modern graphical workstations. Developers typically use modern integrated development environments (IDEs) and make use of a broad cross section of developer tools to manage their development activities. Providing tools that run in the native environment where developers are trained reduces training time for new team members and allows developers to spend less time switching between PC and mainframe environments.

- **Development flexibility and tool range.** The mainframe architecture can impose limitations on developers that can reduce productivity. Moving to contemporary programming environments (e.g., Visual Studio or Eclipse) removes these limitations and also makes available a wide range of tools for developers to use in their development process.
**Micro Focus Enterprise Developer**

The Micro Focus Enterprise Developer product line allows developers to work in their preferred IDE, whether Visual Studio or Eclipse, to author and maintain mainframe COBOL applications. The product line supports single developers and team workflows and brings to bear maturity gained over a decade of research and development. Support for multiple IDEs and the ability to produce COBOL .NET or JVM runtime code means that developers from either the Microsoft or the Java ecosystem can feel comfortable in the development environment as they potentially acquire COBOL skills. IDC research typically finds that a significant proportion of COBOL developers today have already learned newer programming languages in addition to COBOL and thus are likely to have existing familiarity with either Visual Studio or Eclipse, if not both.

**Testing Unplugged**

As a result of portfolio assessment, weaknesses or bottlenecks in service delivery rates, application quality, QA process flexibility, or testing costs may point to a potential requirement to overhaul how core application testing is undertaken.

Unit, functional, and even system testing can provide significant benefits when unleashed from mainframe constraints. Application testing can potentially provide even greater benefits because once development is moved from the mainframe, the entire testing process can be conducted on commodity hardware, taking advantage of a broader variety of tools available for distributed systems. Testing is an integral workload in application development that can tie up significant system capacity, especially during application stress testing and application performance testing.

**Benefits of Off-Mainframe Testing**

Performing mainframe application testing off the mainframe can yield the following advantages:

- **Cost savings.** Reducing mainframe MIPS allocated to testing can be significant, especially in performance tests, which can absorb significant system capacity potentially impacting other mainframe workloads. The episodic and transient nature of performance stress tests can lead to oversizing of mainframe resources and overpaying in software license costs. These costs can be significantly mitigated when testing is offloaded to commodity hardware.

- **Code quality.** The ability to use substantially less expensive hardware resources can also permit deeper and richer testing and expanded test plans, leading directly to higher quality code.
Shorter testing cycles. Taking advantage of large commodity server farms can also lead to significantly shortened testing cycles, and greater resources can incrementally be applied to perform parallel automated testing execution. Overall gains in enterprise agility are thus key benefits of testing applications on distributed platforms.

Easier recruitment. Testing activities offloaded from the mainframe can make it easier to attract and recruit testers and reduces platform training needs considerably.

Richer toolset. Testing on distributed platforms takes advantage of the widest range of testing tools available on the market today, allowing improvements in testing technologies to come quickly to the aid of valuable COBOL applications.

Micro Focus Enterprise Test Server
The Micro Focus testing functionality can be brought to bear on COBOL applications running on modern commodity hardware to support the modernization effort in the enterprise.

Micro Focus Enterprise Test Server provides a mainframe test execution environment under Windows that enables application teams to execute key functional and system test activities free from the cost or constraints of the mainframe environment. This allows testing to scale and flex according to demand and also provides a suitable environment to test broader composite applications including COBOL mainframe back ends and Java or C# interfaces.

The Silk testing product line from Micro Focus is one of the leading testing solutions in the industry today. Silk includes functionality to manage and control the entire testing process, including reporting functionality, advanced manual execution planning, performance testing, and stress testing. The products support data-driven continuous testing, powerful tech-domain scripting, and automated regression testing of enterprise applications.

Deployment Downsizing
In addition to offloading development or testing activities from the mainframe, portfolio assessment results illustrating high ongoing deployment costs, lack of application flexibility, slow delivery of application changes, or the inability to embrace lower-TCO platform opportunities (such as those afforded by later hardware releases, such as zEnterprise) can suggest the need to look at the deployment strategy.

Nowadays, an enterprise can potentially host parts of the application platform to inexpensive commodity hardware. Such rehosting or consolidation can multiply savings ensuing from the hardware systems because it can potentially allow an enterprise to eliminate elements of expensive platform costs and because the application runtime is usually one of the most resource-consuming workloads.
Harnessing Modern System Cost Curves

Over the past decade and a half there has been a steady shift from big iron systems to commodity systems based on Intel's x86/64 and other commodity architectures. These systems are available in a highly competitive market and are supplied by a variety of large vendors offering a rich range of system configurations and service levels. The competitive nature of this commodity market has meant that prices are extraordinarily close to manufacturing costs, leaving vendors narrow margins, which they have become accustomed to operating in. Due to competitive commodity systems, enterprises are thus afforded a highly cost-effective hardware environment that evolves rapidly, gated by the evolution of chip densities and the march of Moore's law. Such systems contrast with proprietary mainframe systems that operate at a higher level of cost and service, potentially providing higher levels of reliability, availability, and security. Tellingly perhaps, the latest range of IBM mainframe technology, the zEnterprise platform, comprises not only traditional z Series chip architecture but also recent releases (e.g., zBX and z114) that include commodity platforms that run AIX, Linux, and Windows, thereby providing that flexibility of choice within a consolidated enterprise server environment. Ultimately, platform preferences are a function of enterprise skill sets and experience. Most large enterprises run diverse shops where mainframes occupy a permanent place. Midsize enterprises gravitate toward reductions in complexity and operational cost to compete with newer members of their peer groups. Older midsize enterprises that automated early and installed mainframes have been the most likely candidates to optimize production workloads using commodity hardware.

New Application Deployment Options

One important likely outcome of an application portfolio analysis is that for stable, vetted server application logic, the problem lies in the deployment hardware and operating system. This can be the case especially for applications hosted on mainframe systems where the mainframe is retained for the exclusive need of a collection of custom-developed COBOL applications. A redeployment option that can relocate the business logic to less expensive Windows or Unix systems can often save an enterprise millions of dollars in maintenance costs, free up mainframe production resources for more critical system expansion, or provide a flexible and cost-efficient replication environment for organizations that need to expand their workload production footprint and want to avoid new mainframe expenditure.

Such deployments may selectively bring various mainframe workloads to less expensive hardware depending on the scope of the project. In many cases, separate decisions can be made with respect to mainframe databases and mainframe application logic. Thus, mainframe databases, while complex to move and require an established data management practice off the mainframe, can also be hosted off the mainframe for additional cost savings in mainframe hardware, software licenses, and maintenance. Alternatively, the databases can be maintained on the mainframe while execution logic that has been moved to commodity platforms can access mainframe databases through distributed data access middleware. The scope and the boundaries of application modernization are normally determined in the analysis phase of the modernization initiative, and an assessment of the enterprise's competencies in the various migrated workloads on nonmainframe platforms is typically part of the analysis. Most enterprises
today run large nonmainframe data management shops, and the decision to move data management often rests on the complexity of the task, the need to redeploy mainframe staff, and an assessment of whether the needed service levels can be attained outside of the mainframe environment.

**Deployment Agility**

Other than savings in the costs of running and maintaining mainframe hardware, the key benefit in deployment on commodity hardware is the ability to respond to changing workload needs more quickly. This deployment agility allows inexpensive new resources to be inserted into the overall system capacity with minimal resources. Additionally, a competitive hosting and infrastructure cloud services market is available for enterprises that wish to take advantage of off-premise resources for further expansion of their workloads. The incrementality of cost and deployment and the elasticity of cloud infrastructure that can act as the deployment environment for existing COBOL applications can go a long way in moving enterprises toward a more agile world.

**Micro Focus Enterprise Server**

A solution to support this scenario is available from Micro Focus in the form of Micro Focus Enterprise Server, which runs on x86 and other commodity hardware and has the benefit of over a decade’s worth of R&D and the referencing of large-scale customer deployments. Micro Focus Enterprise Server supports core mainframe application languages, mainframe communication interfaces, online and batch infrastructure, and access to mainframe databases both on the mainframe and as necessary if these databases are themselves rehosted.

**FUTURE OUTLOOK**

Legacy application evolution has to look to a future in which development skills in legacy programming languages such as COBOL are dwindling. While programmers can be trained on older languages, the costs of training are too high, and few are willing to learn such languages because it will afford them little leverage in the broader application development market.

**Preparing for the Clouds**

Modern architectures based on Web/REST styles of application interaction are more accommodating than ever to encapsulation of existing business logic in its native quiescent state. SOA initiatives undertaken by large organizations set the pace for the Web/REST architectures of today and began the much-needed process of encapsulating existing business logic modules or even whole systems with appropriate wrappers that rendered such systems interoperable in the modern world. SOA brought to the foreground the value of existing applications but did not address the burdens of maintaining business logic. Additionally, the evolving acceptance of cloud infrastructures or of cloud application platforms for growing sections of the enterprise application portfolio, while still in its infancy, suggests a decreased concern for the exact nature of the platform as long as functional fidelity is maintained. In other words, cloud services are gradually habituating enterprises to a modality of operation.
where the service-level agreement (SLA) struck is more important than the exact nature of platforms behind it. In moving applications to the cloud, enterprises may demand functional fidelity but not necessarily an identical platform.

**The Coming Mobile Client Platform Revolution**

Modern back-end systems are architecturally isolated from the type of client front-end components that access them. This, however, does not mean that transformations in client devices will not have an impact on back-end systems. We have seen in the past decade how the Web architecture has emerged to be a dominant front-end model for users, driving many back-end systems to be streamlined or encapsulated as Web services to support the new compositional paradigm of application structuring. Today's mobile revolution is leading to continued changes in native front-end user interaction models through the inclusion of touch interfaces and offline application designs. The mobile front-end architecture as well as the evolving mobile Web will put further pressure on back-end systems, requiring continued work on modularity, encapsulation, and separation of functions into self-contained work units that can be consumed by smaller, task-oriented applications. The need for agility in development and deployment will be highly elevated as occasionally unpredictable transformations are needed to existing applications.

**CHALLENGES/OPPORTUNITIES**

The code preservation approaches highlighted in this white paper are appropriate for a range of enterprise scenarios. However, organizations contemplating these approaches should consider the following challenges.

**The Shrinking COBOL Developer Pool**

Legacy application evolution has to look to a future in which development skills in legacy programming languages such as COBOL are dwindling. While programmers can be trained on older languages, the costs of training are high, and few developers are willing to learn such languages because it will afford them little leverage in the broader application development market. As the pool of COBOL developers shrinks, those willing or able to work in COBOL will begin to demand higher compensation. While there are chronic shortages of developers for almost every platform or programming language, the COBOL developer shortage is mitigated by organic shifting of applications from mainframes to other platforms (for both development and deployment) and by delayed retirements due to the recession. IDC expects the pain points around COBOL to build up through the rest of the decade and reach their peak sometime during the next decade. This leaves a considerable period of time to leverage existing COBOL assets through the use of techniques outlined in this white paper.

**Architectural Dead Ends**

Not all code is suited for long-term evolution. Certain applications may be designed around older architectural concepts that may no longer be suited for new requirements. It is important for organizations to make the right calls on the expected
longevity of code assets during the analysis phase of the modernization effort. An architectural assessment should be part of the overall analysis phase, and applications that do not conform to the needed architectural standards may have to be earmarked for replacement with purchased or newly developed applications on modern application platforms.

CONCLUSION

In broad terms, application modernization can be seen as comprising four high-level stages: analysis, development, testing, and deployment. A thorough analysis phase can result in the classification of legacy application assets according to any number of appropriate taxonomies and help pinpoint which modernization activities will yield the greatest return. High-value applications that are stable and yet require continual maintenance can be developed, tested, and even deployed on modern application platforms without rewriting the COBOL code. Such approaches can yield significant development and delivery cost savings, enhance developer productivity, and potentially make it more attractive for developers to be trained on COBOL using familiar modern developer tool chains such as Visual Studio and Eclipse.

In fact, with such available approaches and a contemporary model into which core COBOL business systems can be transformed, the term "legacy" as it pertains to these systems is no longer accurate. As CIOs who run their business on COBOL have indicated to IDC on more than one occasion, "These applications are not legacy; these are my core business."

LEARN MORE

Related Research

- Worldwide Development Languages, Environments, and Tools 2011 Vendor Shares (IDC #235129, June 2012)
- Going Mobile: The Coming Convergence of Front-End Application Platforms and Ecosystems (IDC #234000, March 2012)
- Application Development Trends: Developer Ecosystems in Flux (IDC #DR2012_BB1_AH, Directions 12 Presentation, March 2012)
- IDC Predictions 2012: Application Development & Deployment (IDC #WC20120117, January 2012)

Copyright Notice

External Publication of IDC Information and Data — Any IDC information that is to be used in advertising, press releases, or promotional materials requires prior written approval from the appropriate IDC Vice President or Country Manager. A draft of the proposed document should accompany any such request. IDC reserves the right to deny approval of external usage for any reason.

Copyright 2012 IDC. Reproduction without written permission is completely forbidden.