ESP-IJACT
ESP International Journal of Advancements in Computational Technology
ISSN: 2583-8628 / Volume 3 Issue 2 April 2025 / Page No: 10-17
Paper Id: IJACT-V3I2P102 / Doi: 10.56472/25838628/IJACT-V3I2P102

Original Article

DevOps for Mainframe Environment, Accelerating Legacy Software Delivery through Continuous Integration and Deployment

Bhargav Kumar Konidena[1], Swetha Ravipudi[2], Abdul Samad Mohammed[3]
[1]*Vintech Solutions Inc, USA,*
[2]*Lucid Motors, USA.*
[3]*Dominos, USA.*

Received Date: 14 February 2025 Revised Date: 25 March 2025 Accepted Date: 08 April 2025

Abstract: *Mainframe systems remain the backbone of critical industries such as banking, healthcare, and government, handling vast amounts of transactional data with unmatched reliability. However, traditional mainframe software delivery methods often manual and sequential—lag behind modern DevOps practices, creating bottlenecks in agility, deployment speed, and innovation. This research explores the integration of DevOps, Continuous Integration (CI), and Continuous Deployment (CD) into mainframe environments to bridge this gap. By analyzing challenges such as legacy architectures, cultural resistance, and compliance constraints, we propose a structured Mainframe DevOps Modernization Framework that incorporates toolchain modernization, automated testing, and deployment strategies like blue-green and canary releases. Case studies and industry surveys demonstrate that DevOps adoption can reduce deployment times by up to 60% while improving code quality by 40%. The study also highlights future research directions, including AI-driven automation, Mainframe-as-a-Service (MFaaS), and quantum computing's potential impact. This work provides actionable insights for organizations seeking to modernize legacy mainframe systems while maintaining their stability, security, and scalability in a rapidly evolving digital landscape.*

Keywords: *Mainframe DevOps, Continuous Integration (CI), Continuous Deployment (CD), Legacy Modernization, Test Automation.*

I. INTRODUCTION

Critical operations that span the spectrum of industries including banking, insurance, healthcare and government have relied on mainframe computers to address their challenges for decades[1]. Mainframes still handle over $7 trillion in credit card transactions, yet despite the growth of distributed and cloud-native computing models, mainframes process 68 percent of the world's production IT workloads. However, other than their unmatched reliability, scalability and transaction processing power [2], organizations still rely on it. However, mainframe application maintenance and enhancement processes have generally been slow, manual, and resistant to change, and with digital economy software delivery expectations so rapid, this stark contrast produces a threat to business.

The gap between traditional mainframe development and the modern agile practices has grown wider, as businesses are under increasing pressure to get the innovation out faster [3]. Cloud native environments adopt Agile, DevOps, and Continuous Integration/Continuous Deployment (CI/CD) pipelines to accelerate release, whereas mainframes legacy systems still rely on repetitive and complicated, long release cycles and manual testing [4]. The IBM's 2023 "State of the Mainframe" report asserts that more than 60 percent of mainframe updates take place through manual or sequential process, and cycle times are, well, three to five times slower than for agile cloud native systems [5]. As a result of this technological inconsistency, development bottlenecks, ineffectiveness in meeting market demands, and increased operational risk emerge. Figure 1 shows how much faster cloud-native compared to mainframe, while illustrating how legacy systems struggle to achieve modern delivery expectations.

As a natural outcome of some of the tenets of DevOps - collaboration, automation, continuous feedback , mainframe software delivery modernization is a transformative opportunity [6]. When DevOps practices are embraced in a mainframe setting, meanwhile, substantial operational improvements can be gained, including a 60-percent decrease in deployment times and a 40-percent increase in code quality, as quipped by a 2022 BMC Software survey [7]. Additionally, CI practices can integrate impacts for early detection of and fixes to integration issues as well as CD that can automate release in a very timely fashion minimizing downtime, reducing human error and delivering more frequent updates.

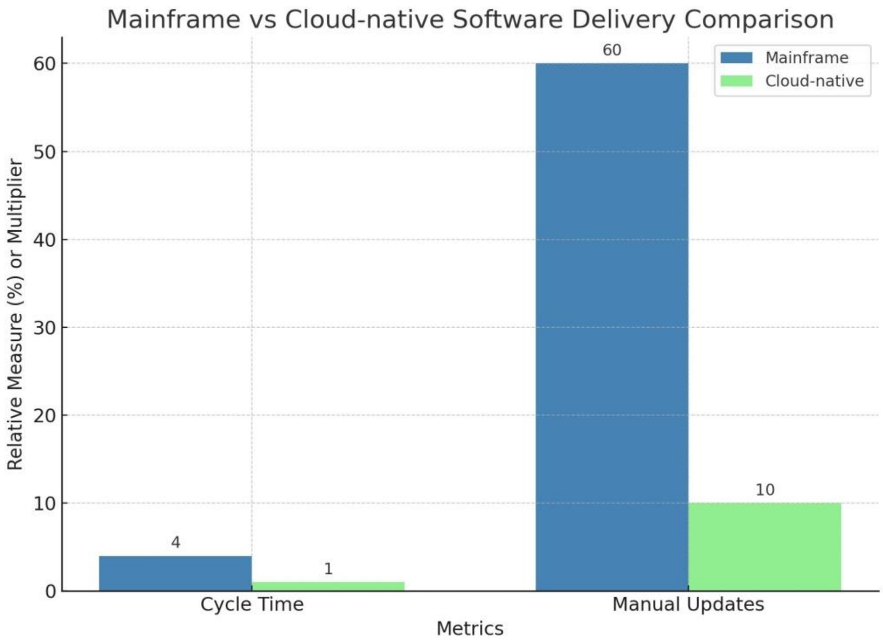

Figure 1: Mainframe vs Cloud-native Software Delivery Comparison

While implementing DevOps in the mainframe scenario is not an easy task, despite that. Transforming is hindered by legacy technologies, cultural resistance by traditional development teams and the need for extreme compliance along with decades of technical debt. The main role of mainframes in core business operating also requires stability, reliability, and security must not be undermined. To this effect, organizations have to carefully adapt DevOps principles to the specialization of the mainframe systems, making use of the hybrid approach that keeps the system integrity in order and drives agility and innovation.

Key Objective:

To systematically explore the potential of DevOps in mainframe environments, this research is guided by the following core objectives:

- Compare the classic ways of delivering traditional mainframe software with recent agile and DevOps practices and analyze their bottlenecks.
- Integrate the DevOps principles (of Continuous Integration and Continuous Deployment) into legacy mainframe environment.
- Show the operational challenges and benefits of adopting DevOps in mainframe systems both in terms of deployment speed, code quality and system stability.

II. BACKGROUND

Government cloud deployment is driving IT modernization, improving service delivery, and cost savings. Nevertheless, moving from legacy systems to cloud-native infrastructure is challenged by meeting regulations and safeguarding data [12]. This section takes into account the role of GovCloud platforms, key compliance standards, and security gaps in current security strategies in public sector cloud deployment.

A. Evolution of Mainframe Systems

These are instances of technological resilience and innovation, the history of mainframe computing. Several phases of evolution of mainframes have been passed, each was characterized by progressive increase in processing power, storage capacity and integration capabilities [8]. A table for key milestones in mainframe systems evolution is below.

Mainframer IBM made a revolutionary step forward in its computing with the IBM System/360 in 1964. That it could be employed in wide variety of scientific and commercial applications while keeping abreast with different hardware configurations made it a game changer for such industries as banking, healthcare and government.

The System/360 was very successful and IBM used its success with mainframe technology to improve and further refine it. The microprocessing advancement and improved storage capabilities were brought in by the IBM System/370, launched in the 1970s. This is how mainframes became faster, more reliable, and more powerful and are used as mission critical applications.

Table 1: IBM Mainframe Evolution: 1964 to 2019

Year	Mainframe Model	Key Features	Impact
1964	IBM System/360	Unified architecture, compatibility across hardware models, support for scientific and commercial applications.	Set the foundation for modern mainframe systems.
1970s	IBM System/370	Advanced microprocessing, improved storage, enhanced reliability.	Increased processing power and reliability.
1980s-90s	IBM zSeries	Introduction of parallel processing, improved virtualization, optimized for large-scale enterprise applications.	Introduced virtualization and scalability.
2019	IBM z15	Hybrid cloud support, enhanced security (data encryption), scalability for exponential data growth.	Integration with cloud environments, modern workloads.

Starting from the 1980s and 1990s, IBM's zSeries introduced parallel processing and enhanced virtualization to enable enterprises to utilize IT resources optimally to meet their ever growing data requirements. The zSeries opened the doors for the high performance systems upon which we depend today. Along with the IBM z15 mainframe, released in 2019, this was a big step towards modernization of mainframe technology. It runs in the hybrid cloud, is secure with data encryption and comes with the scalability that is necessary with an exponential growth of data. The z15 is optimized for both the traditional workload as well as the cloud native applications to maintain the mainframes role in modern IT landscape.

B. Types of Mainframe Workloads

High reliability, security and uptime are essential in many industries that use mainframe systems. Table 2 shows some of mainframes' most common workload categories. They reveal the continuous need for mainframes in today's IT infrastructure. While cloud native apps continue to grow, the mainframe has not gone away, essential because it allows you unmatched stability, processing power and transaction integrity [9].

Mainframe to cloud technology integration is seamless as IBM z15 has the hybrid cloud capabilities and organizations can combine the best of both - for mission critical workloads on mainframes and for dynamic, cloud native application on cloud.

C. Introduction to DevOps, CI, and CD

During a fast changing software development landscape, the traditional approaches for software delivery no longer cover the growing speed, flexibility and quality requests [10]. As a result, DevOps, a pioneering method of bringing Development (Dev) and Operations (Ops) teams together for improved operations, faster software delivery, and higher quality of products has emerged. Three core principle of DevOps are Collaboration, Automation and Continuous Feedback. These concepts are based on the idea of speed and efficiency when developing software, in the way to deliver faster and better quality of software.

a) Core Concepts of DevOps

- *Collaboration:* Until recently, the developers and operation teams were operating in silos, and most of the time this has resulted in inefficiencies and delays. DevOps brings teammates together across teams development, operations, security, and others. The constant communication ensures alignment and less misunderstandings and overall productivity [11].
- *Automation:* DevOps is synonymous with automation. It deploys the process of eliminating the manual and repetitive tasks like code integration, testing and deployment. Automating such tasks grants teams the ability to cut down on human errors, as well as enhance consistency, and save development time spent in deploying rather than in developing new features [12].

Table 2: Mainframe Workloads: Core Processing Types and Modern Integration

Workload Type	Description	Example Use Cases
Transactional Processing	Mainframes excel in processing high-volume, low-latency transactions with accuracy and reliability.	ATM networks, credit card payments, insurance claims processing.
Batch Processing	Ideal for handling large datasets processed in scheduled intervals, such as payroll or financial reconciliations.	Payroll processing, financial reporting, data migration.
Enterprise Resource	Mainframes run integrated ERP systems that manage key	SAP, Oracle ERP systems for

Planning (ERP)	business processes like finance, HR, supply chain, and customer relations.	large organizations.
Data Warehousing	Mainframes are used to aggregate, store, and analyze vast amounts of data for business intelligence purposes.	Large-scale data analytics, historical data analysis.
Cloud Integration and Hybrid Cloud	Mainframes work with cloud technologies to create hybrid environments that combine the strengths of on-premises mainframe systems and the scalability of the cloud.	IBM Z Hybrid Cloud solution.

- *Continuous Feedback:* Over the life cycle of the software, qualities such as reliability and the quality of software are maintained with the help of continuous feedback. DevOps enables early bug, security vulnerabilities and performance issues with continuous integration and deployment (CI/CD). Giving teams real time feedback allows them to make quick changes to software quality and shorten time to market.

b) DevOps Lifecycle: A Brief Overview

It is a continuous process from the start of software development to the end of it, thus integrating development, testing, deployment, and operations in it. It is meant to improve efficiency, speed and collaboration [13]. Here are the stages that make up the DevOps lifecycle shown in table 3 below:

c) DevOps Lifecycle in Action

- *Planning:* During the planning, a team should define, set objectives and plan the iterations of the project. In this phase, agile principles make sure that this phase remains flexible for receiving ongoing feedback as well as changes to the requirement.
- *Development:* In this phase, developers write code based on the defined requirements. Through collaboration with operations teams, code is designed to work seamlessly within the deployment environment.
- *Run, Deploy:* Once ready with the code, it is then run and deployed into a host. The process of building is automatically created by automated build tools and run quickly and without errors.
- *Automated tests:* The code is tested for quality standards. The testing involved here involves unit testing, integration testing and other applicable tests to ensure the code works properly as it is supposed to.
- *Preparing:* The test prepares the code for release. CD tools, by way of reduction of manual effort and incorporation of an automatic step to push code out to staging or production environments, are continuously deploying.
- *Automated deployment:* The deployment phase is automated, to prevent release process becomes smooth and effective, hence reduce downtime, reduce risks by manual deployment.
- *Run:* Once live the software is operated monitoring its performance and adjusting its behaviour to maintain peak performance. Any issues bypass automated systems and alert teams, where the problems can be quickly solved.
- *Understanding these details empowers monitoring:* Continuous monitoring so that teams can monitor user behavior, application performance and health of system. Monitoring tools to ensure ongoing stability are needed and feedback is needed for improvement of future versions of the software.

In today's business world, revolution driven by DevOps, of collaboration, automation and continuous feedback, has led to the software that was developed, tested, and deployed. Continuous Integration (CI) and Continuous Delivery (CD) of the Dev Ops cycle speeds up, increases reliability and improves quality of the most recent software delivery. In addition to the fact that these principles improve the overall efficiency of the software development, they make it possible for organizations to maintain their agility and respond to the market demands on top of that.

III. LITERATURE REVIEW

Table 3: DevOps Lifecycle Stages and Key Activities

Stage	Description	Key Activities
Planning	Define project goals, requirements, and timelines. Agile methodologies are often employed here.	Requirement gathering, sprint planning, goal setting.
Development	Code is developed and new features are created. Collaboration between Dev and Ops begins here.	Feature development, coding, unit testing.
Build	Compile code into deployable packages. Automated build tools are used to streamline the process.	Code compilation, dependency management.
Testing	Code is automatically tested to ensure quality. This includes unit tests, integration tests, etc.	Unit testing, integration testing, performance testing.
Release	After successful testing, code is prepared for release. Continuous Deployment tools automate this.	Release preparation, staging, configuration management.

Deploy	Code is deployed to production environments. Automation ensures minimal downtime and risks.	Deployment to production, post-deployment testing.
Operate	The software is monitored for performance, security, and stability. Operational issues are tracked.	System monitoring, performance analysis, incident management.
Monitor	Continuous monitoring provides feedback on system health and user behavior. Issues are addressed.	System health monitoring, user feedback collection.

There's been recently a lot of research about embracing, moving to, or adopting, modern development methodologies such as DevOps and Continuous Integration and Continuous Delivery (CI/CD) in a traditional environment, such as mainframe systems. Several studies and case studies have been undertaken with respect to the issues that is the opportunities, solvable and best practices for the modernisation of mainframe operations, and the legacy systems to be supported. This section rounds up and examines the know literature in this regard as done in the context of DevOps integration and application in mainframe modernization and identifying and dots empty.

A. Research on Mainframe Modernization

As these industries had no demand for any traditional multi tier Web application based development, most of the legacy development teams could spend their time and resources in building business critical applications exposed through mainframe systems. While the technology is not outdated, the infrastructure used with it is, as is the case today, modern: agile, cloud native infrastructure for the modern version of software development. Particularly interesting is investigations on mainframe modernization, as an approach to increase the mainframes' ability to integrate to the current IT environment in the ways that parallel the advances in the areas of scalability, security, reliability of the latter without impairing them.

a) Mainframe as a Service (MaaS) and Hybrid Cloud Solutions:

Researcher [14] examined how mainframe systems can be incorporated into a hybrid cloud, allowing expedience for companies to operate a legacy system while taking advantage of the scalability present in the cloud. It's rather crucial, their findings stressed, for a hybrid cloud architecture that mingles mainframes where they shine with cloud platforms — for instance IBM's z15 hybrid cloud integration . Different from the traditional large scale mainframe modernization such as in the OTF that only focused on the modernization of functionally outdated applications, Mainframe modernization is now being seen more as a gradual, step by step approach using a mix of cloud native and legacy components giving maximum efficiency.

b) Modernization via Virtualization and Containerization:

Modernization of the mainframe workloads will include virtualization and containerization. [15] described the adoption of virtualization technologies such as z/VM and Linux on Z that allowed companies to optimize resource allocation, do a good job of managing workloads, and allow for modern software development practices. By allowing multiple and isolated environments on a single physical system these technologies make mainframes more compatible with DevOps practices.

i) Integration of DevOps with Legacy Systems:

There has been increased research into DevOps adoption in legacy mainframe systems, and a number of studies have shown that modernizing the practices of large organizations is not a simple task. Following that, researchers [16] explored how DevOps practices could be used for the mainframes – with tools like Jenkins, Git and Docker – and also the role of the mainframe as a Service (MaaS) platforms in closing the gap between the legacy infrastructure and the current software delivery pipeline. Despite those, many organizations still struggle with cultural resistance to change as well as limitations of legacy toolchains.

c) CI/CD in Mainframe Environments:

The CI/CD in mainframe environments [17] was focused. It also showed how IBM UrbanCode and TeamCity can automate mainframe build, test, and deploy processes to enhance speed and reliability of delivering of mainframe systems. Since then, there has been a lot of advancement, but CI/CD mainframes is still complex, and in many cases that leads to org's not being able to fully integrate CI/CD into there mainframe based systems because of technical debt and limited tool support.

B. Limitations and Gaps in Prior Work

While existing research and case studies provide valuable insights into the potential for DevOps in legacy environments, several limitations and gaps remain:

a) Cultural Resistance:

There is therefore a lack of studies addressing the cultural resistance from within the organizations, and especially in traditional mainframe developers to adopt DevOps practices [18]. Relationship between DevOps and its different components, in a sense, is neglected in literature as there is a lack of focus on a shift in the mindset required for adoption of a DevOps.

b) Tool Integration Issues:

An issue present in existing research is that modern DevOps tools are not integrated with the legacy mainframe systems[19]. Largely, due to existing legacy systems that are mostly incompatible with the cloud native DevOps tool and as a result, make it very difficult to fully automate build, testing and deployment processes.

c) Security and Compliance Constraints:

Most DevOps efforts in mainframes are faced with security and compliance barriers in the contexts of finance and healthcare industries in particular. However, there isn't much in the way of research of how organizations can get the speed of DevOps and adherence to the stringent security requirements of legacy systems[20].

d) Scalability of DevOps on Mainframes:

Currently little has been done in research toward the scaling of DevOps practices for large scale mainframe environments. DevOps is understood but applied to cloud native and distributed systems, whereas mainframes are profoundly different from ones, and scalability issues with automation and integration tooling are yet to be addressed [21].

This literature review explores the rising role of DevOps in the mainframe environment and reports on advances in integration of modern practices to mainframe systems. Yet, there are still challenges, including integration of the tools as well as cultural shifts and making security and compliance. Thus, future research should target addressing these gaps to deliver more actionable insights to organizations that are looking to modernize their mainframe operations, while still retaining the stability and reliability they have depended on.

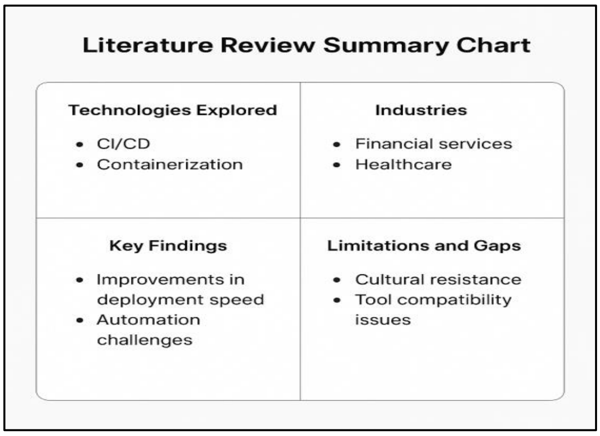

Figure 2: Literature Review Summary Chart

IV. CHALLENGES IN ADOPTING DEVOPS FOR MAINFRAMES

There are notable issues in adopting DevOps practices in any legacy system environment, especially mainframe environments as they have their own distinct characteristics: organizational dynamics, regulatory environments, legacy system, in addition, which makes meeting the objective more challenging [22]. The potential benefits of DevOps for speeding up software delivery and improving code quality are apparent but it is not easy to realize these potential benefits without addressing a number of technical, organizational, and process obstacles of DevOps in mainframe contexts.

A. Technical Challenges

a) Legacy Languages (COBOL, PL/I):

Mainframe Legacy languages like COBOL and PL/I are built to use in building mainframe applications for older hardware and workflow. However, support for these modern development practices such as automation, control version and CI/CD don't exist in these languages and developers need to integrate them into DevOps tools. It also further widens the gap with limited expertise among the newer developers in the team that complicates the efforts to modernize and adopt continuous delivery practices.

b) Lack of Modern APIs:

Initial mainframes were closed systems and were very little intertwined with the outside world. The absence of modern APIs makes it difficult to integrate the mainframe application with the cloud native services as well as the DevOps tools like Jenkins, Git or Docker. This makes it difficult to automate workflows, extend functionality, work natively with broader enterprise systems, or embed into other systems seamlessly.

c) Complex Monolithic Architectures:

Most mainframe systems are large tightly coupled monolithic systems. But small changes can have large system affect and make testing and deployment risky and slow. These architectures are so rigid, that continuous integration and agile delivery are increasingly difficult to implement without a great amount of planning and refactoring up front, which is not so DevOps friendly.

B. Organizational and Cultural Challenges

i) Resistance to Change:

Cultural There is still much cultural resistance at large. Waterfall is also pervasive throughout mainframe teams, often focused on stability, risk aversion. When it comes to moving to DevOps from these more bureaucratic approaches of development, there is a real need for a very deep cultural shift that requires strong leadership, ongoing education, and incentives around a new set of practices.

ii) The Dev Ops Teams are Siloed Between Dev and Ops:

Traditionally, developers and operations teams work separate from each other, miscommunication not only happens but also wastes resources. Close collaboration as well as sharing of ownership are called for from DevOps between these groups. Entrenched silos would mean time, active efforts and leadership support needed to break down and change the established workflows and to rebuild trust and coordination.

C. Process and Governance Challenges

a) Compliance-Heavy Environments:

Industries Mainframe systems are heavily depended on in industries such as finance, healthcare and government and in order to comply with their strict compliance, these industries have strict compliance requirements placed on their mainframe systems. However, DevOps is about rapid iterations but compliance requires proper documentation, audits and risk controls. This presents a challenge however as these two priorities have to be reconciled while allouating automation, while not jeopardising regulatory integrity.

b) Long Approval Cycles:

Mainframe changes of production typically need approval processes which take forever to complete and deploy. While the DevOps tools are present, such long cycles are unable to enable fast iteration. To be in line with DevOps principles, organizations need to streamline the approval workflows, introduce automation where they can and have smaller, manageable change sets.

While there is much to be rewarded in adopting DevOps within the mainframe environment, there are challenges. However, real obstacles are created by legacy technologies, cultural inertia, and tight governance processes. The success lies in a proper approach that is able to respect the needs stability of mainframes, while adopting modern development practices to foster agility, collaboration, and innovation. By adopting a modern approach towards their mainframe operations, organizations can make their business ready for the digital economy.

V. MAINFRAME DEVOPS MODERNIZATION FRAMEWORK

DevOps practices for mainframe system must be oriented by a structured approach to modernize both mainframe system technology stack and the organizational process. Transitions will be successful if it also means a technical integration of modern tools and methodology but also if it can lead to strategic and cultural reorientation to be in line with the business limitations and business targets. The main points of modernizing environment frameworks for DevOps adoption are described in this framework.

A. Strategic Planning for DevOps Adoption

a) Assessment of Current State:

The first step of adopting DevOps in the mainframes is the assessment of its current state of mainframe systems. To do this, you have to know the architecture of the organization, how it is developed and deployed across the whole organization, what technologies are used, what the culture of the company is like, etc. This is because it is necessary [23] to identify the kind of the pain points in the existing workflow (slow release cycles, the need for manual testing, fragmented toolchain), and assess the degree of readiness of the legacy systems for the traditional DevOps. It enables organizations to understand exactly where to concentrate efforts so they can modernize accordingly.

b) DevOps Adoption Needs Stakeholder Buy-in:

For DevOps adoption to be successful, each key stakeholder across an organization, including developers, operations teams, *and business leads* and compliance officers need to be actively committed and involved. If buy in from stakeholders is not secured, you won't have the necessary resources to work with, you won't have common goals, and change management will prove troublesome. Education, communication, and pilot projects are valuable means for engaging stakeholders early in the process and demonstrating a few of DevOps' benefits including a shorter time to market, better system stability, and better collaboration (between the devs and the ops folks). The more likely organizations are to effectively transform, the more likely they will have a shared understanding of the business of DevOps.

B. ***Modernization of Toolchains***

a) SCM Modernization:

Git on z/OS:Replacing traditional source control management (SCM) tools with modern, flexible systems such as Git is one of the largest shifts in modernizing mainframe development. With tools like Git on z/OS developers can get a version control system widely used within industry, allowing for better collaboration, branching and version tracking. This modernization allows teams on the mainframe to easily integrate with these broader DevOps workflows and work in parallel with those that are managing cloud native applications. Having Git repositories on z/OS makes it possible to consolidate the SCM practice across the mainframe and other environments and proceed with consistent workflows for the entire development lifecycle.

b) Key step of DevOps mainframe modernization for CI/CD pipelines:

Jenkins, GitLab CI, UrbanCode and BMC Tools. A build, test and deployment process can be automated using Jenkins, GitLab CI, UrbanCode, and even BMC tools that help with faster and more reliable delivery cycles. They are used to help automate how the movement of code can go through development to production as efficiently and repeatably as possible. Integrating CI/CD tools in mainframe workflows may require tweaks to work around the peculiarities of the mainframe environment, and these may involve creating special configurations of tailored versions of the build tools specific to z/OS and other mainframe facilities and handling large batch processes. It is about reducing manual intervention and to some extent automate deployments.

C. Test Automation on Mainframe

a) Automated Unit Testing Tools (e.g., Topaz for Total Test):

Test automation is a key component of DevOps, helping ensure that code changes are properly validated before they reach production. For mainframes, tools like Topaz for Total Test provide an environment for automated unit testing, supporting modern testing practices such as Test-Driven Development (TDD) and Continuous Testing (CT) [24]. These tools are designed to handle the complexity of mainframe applications, including legacy languages like COBOL and PL/I, and provide real-time feedback on code quality. By automating unit tests, organizations can accelerate the testing process, reduce the risk of defects in production, and improve overall code quality.

D. Build Automation

a) Gradual Integration of Build Scripts and Task Runners:

The ability to automate the process of compiling and packaging software to be installed on a target system is crucial for its ease of use. In most mainframe environments, older manual build processes are slow and prone to error. Organizations should modernize by bringing the execution of build scripts and task runners together with the tools used for compiling COBOL programs, packaging executable files, and creation of deployment artifacts. There are tools like Jenkins and UrbanCode that can help justify the definition and management of build pipelines, and these tools are used to guarantee that the same automated build pipeline is used across all environments. As the script itself evolves and gets more business value these scripts will also evolve to handle more complex tasks and ultimately integrate with other stages of the CI/CD pipeline.

E. Continuous Deployment Approaches

a) Green-Blue Deployments, Canary Releases on Mainframes:

Continuous Deployment (CD), along with infrastructure as code (IaC) is one of the big ticket items of DevOps where code changes are deployed in a continuous basis to the production environment. Building CD on mainframe is difficult due to the nature and size of mainframe applications [25]. However, these techniques (green-blue deployment, canary release) can be adapted for reducing deployment risk and improving release management for mainframe systems.

b) Green-Blue Deployment:

In this case, two equivalent environments are created (green and blue). Traffic is switched from green environment to blue environment after the new version of application is released to the blue environment and after fully verified. This guarantees that in case an issue occurs, the green environment is still available for rollback.

A canary release (also called roll out release) is a small % of users get the new version of the app and if all goes well, it's slowly deployed to other users. Therefore, if there is an issue, then only a small portion of users is affected at a time.

Integrated into the CI/CD pipeline, both approaches can be automated using tools, so that mainframe applications are rapidly deployed with little risk to downtime or service disruption.

Table 4: Devops Tools Landscape for Mainframe

Category	Tools	Description
Version Control	Git on z/OS, Endevor, ChangeMan	Modern version control systems for mainframe code integration and collaboration.
CI/CD Automation	Jenkins, GitLab CI, UrbanCode, BMC Control-M	Tools to automate code integration, testing, and deployment pipelines.
Build Automation	Jenkins, UrbanCode, IBM Build, Make	Tools to automate the build process for mainframe applications.
Test Automation	Topaz for Total Test, IBM Rational Test	Tools for automated testing of mainframe applications, including unit and functional testing.
Deployment Automation	UrbanCode, z/OS Deployment Manager	Tools for automating the deployment process, including green-blue and canary releases.

This framework sets the stage for organizations who are interested in modernizing the mainframe systems using DevOps practices. With support of key aspects of strategic planning, toolchain modernization, test and build automation and deployment approaches, mainframe teams can accelerate their software delivery cycles while ensuring stability and security of their legacy systems.

VI. PROPOSED MODEL FOR MAINFRAME DEVOPS ADOPTION

This model provides a structured approach to integrating DevOps practices within a mainframe environment, aimed at accelerating legacy software delivery through Continuous Integration (CI) and Continuous Deployment (CD).

A. Assessment and Planning

- *Evaluate Current State: Assess the mainframe infrastructure, identifying pain points and legacy challenges.*
- *Stakeholder Engagement:* Align development, operations, and management teams around DevOps objectives.
- *Risk Management:* Address compliance, data privacy, and operational risks in modernizing legacy systems.

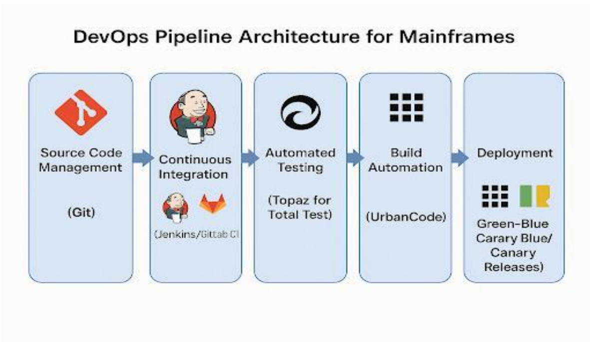

Figure 3: DevOps Pipeline Architecture for Mainframes

B. Phased Implementation

a) Phase 1: Toolchain Modernization

- *Integrate modern tools (e.g., Git on z/OS, Jenkins, UrbanCode) for version control, CI/CD, and testing.*
- *Automate the build and testing process with modern mainframe tools (e.g., Topaz for automated testing).*

b) Phase 2: Process Automation

- *Implement build automation tools to streamline code compilation and packaging.*
- *2. Adopt deployment strategies like blue-green and canary releases to reduce downtime.*
- *Integrate automated tests into the CI/CD pipeline for consistent quality assurance.*

c) Phase 3: Cultural and Organizational Transformation

- *Foster collaboration between development and operations teams.*
- *2. Appoint DevOps champions to drive the transformation and encourage cross-functional knowledge sharing.*
- *Provide training on DevOps principles and tools.*

d) Phase 4: Continuous Improvement

- *Monitor KPIs (e.g., deployment frequency, MTTR) to ensure the pipeline's efficiency.*

- *2. Use feedback to continuously refine processes and enhance automation.*
- *Hold regular retrospectives to identify improvement areas.*

C. DevOps Pipeline Architecture

- *Version Control:* Use Git on z/OS for source code management.
- *Automation Tools:*Integrate Jenkins and GitLab CI for continuous integration and deployment.
- *Test Automation:*Incorporate tools like Topaz for automated testing.
- *Deployment:* Use UrbanCode or custom solutions for automated deployment and rollback strategies.
- *Monitoring:* Implement monitoring tools to track pipeline health and optimize performance.

This model enables organizations to successfully adopt DevOps in mainframe environments, overcoming legacy challenges while modernizing processes for faster and more reliable software delivery.

VII. FUTURE RESEARCH DIRECTIONS

A. AI/ML-driven DevOps for Mainframes

*I*nvestigating ways in which AI and machine learning can automate tasks such as anomaly detection and optimization of performance in DevOps pipelines on mainframe.

B. Mainframe-as-a-Service (MFaaS)

Considering how the mainframe capabilities can be offered as a cloud service such that organizations more flexible and scalable solutions.

C. Quantum Computing Impacts

Quantum computing seeks to revolutionize mainframe processes, including data encryption, optimization, and AI-driven work that is more sophisticated.

D. Automation in Compliance and Security

Focusing on finding further automation of compliance checks on mainframe DevOps so that accuracy improves and human error diminishes.

VIII. CONCLUSION

The integration of DevOps into mainframe environments offers a strategic pathway to modernize legacy systems while maintaining their unmatched reliability and security. Despite challenges—such as outdated toolchains, monolithic architectures, and cultural resistance—organizations can achieve significant improvements in deployment speed, code quality, and operational efficiency by adopting a structured Mainframe DevOps Modernization Framework. Key steps include modernizing version control (e.g., Git on z/OS), automating CI/CD pipelines (Jenkins, UrbanCode), and implementing risk-mitigating deployment strategies like blue-green releases.

Industry evidence shows that DevOps adoption can reduce deployment times by 60% and enhance code quality by 40%, bridging the gap between mainframe stability and agile delivery. Success hinges on cultural alignment, stakeholder buy-in, and iterative automation. Future advancements in AI-driven automation, Mainframe-as-a-Service (MFaaS), and quantum computing promise further optimization.

Ultimately, DevOps is not just for cloud-native systems—it is a critical enabler for mainframes to remain competitive in a fast-evolving digital economy. By balancing innovation with legacy constraints, enterprises can transform their mainframe operations into agile, future-ready assets. The journey requires commitment but delivers long-term gains in efficiency, scalability, and business responsiveness.

IX. REFERENCES

[1] S. Varatharajan, S. Santhosh, and A. Subramanian, "Reimagining Insurance: A Strategic Shift from Mainframe Systems to Cloud-Based Operations."

[2] R. Yang and J. Xu, "Computing at massive scale: Scalability and dependability challenges," in 2016 IEEE Symposium on Service-Oriented System Engineering (SOSE), IEEE, 2016.

[3] S. Denning, "Why Agile can be a game changer for managing continuous innovation in many industries," Strategy & Leadership, vol. 41, no. 2, pp. 5-11, 2013.

[4] V. U. Ugwueze and J. N. Chukwunweike, "Continuous integration and deployment strategies for streamlined DevOps in software engineering and application delivery," Int. J. Comput. Appl. Technol. Res., vol. 14, no. 1, pp. 1-24, 2024.

[5] Kyndryl, Kyndryl's 2023 State of Mainframe Modernization Survey Report with a Focus on Government Organizations. [Online]. Available: https://www.kyndryl.com/content/dam/kyndrylprogram/cs_ar_as/moving-off-mainframe.pdf

[6] A. Mehta and P. Ranjan, "The Role of DevOps in Accelerating Digital Transformation," Baltic Multidisciplinary Research Letters Journal, vol. 1, no. 3, pp. 25-35, 2024.

[7] R. Amaro, R. Pereira, and M. M. Da Silva, "Capabilities and practices in DevOps: a multivocal literature review," IEEE Trans. Softw. Eng., vol. 49, no. 2, pp. 883-901, 2022.

[8] A. J. Singh and A. P. Singh, "Evolution of Information Technology: A Comprehensive Review," Asian J. Curr. Res., vol. 9, no. 4, pp. 78-90, 2024.

[9] M. Ebbers, et al., IntrSoduction to the New Mainframe: IBM Z/VSE Basics, IBM Redbooks, 2016.

[10] P. G. Smith, Flexible Product Development: Building Agility for Changing Markets, John Wiley & Sons, 2007.

[11] G. Bou Ghantous and A. Gill, "DevOps: Concepts, practices, tools, benefits and challenges," in PACIS 2017, 2017.

[12] M. Gall and F. Pigni, "Taking DevOps mainstream: a critical review and conceptual framework," Eur. J. Inf. Syst., vol. 31, no. 5, pp. 548-567, 2022.

[13] B. Aiello and L. Sachs, Agile Application Lifecycle Management: Using DevOps to Drive Process Improvement, Addison-Wesley Professional, 2016.

[14] L. Miller and M. S. Raja, "Evaluating Cloud Scalability Post-Migration for Mainframe Workloads: A Comparative Analysis."

[15] L. Parziale, et al., End-to-End High Availability Solution for System z from a Linux Perspective, IBM Redbooks, 2014.

[16] U. Kalesnikau, "The Easy Way to Bring DevOps to Mainframe Software Development," IBA Group, Oct. 28, 2021. [Online]. Available: https://ibagroupit.com/insights/the-easy-way-to-bring-devops-to-mainframe-software-development/

[17] O. Ivanov, "Development of CI/CD platform deployment automation module for group software development," 2021.

[18] M. S. Khan, et al., "Critical challenges to adopt DevOps culture in software organizations: A systematic review," IEEE Access, vol. 10, pp. 14339-14349, 2022.

[19] M. Schwartz, "The Role of DevOps in Legacy System Integration," 2019.

[20] L. E. Lwakatare, et al., "DevOps in practice: A multiple case study of five companies," Inf. Softw. Technol., vol. 114, pp. 217-230, 2019.

[21] S. Sharma, The DevOps Adoption Playbook: A Guide to Adopting DevOps in a Multi-Speed IT Enterprise, John Wiley & Sons, 2017.

[22] A. E. Schaller, DevOps Transformation Challenges Facing Large Scale Legacy Systems, MS thesis, Utica College, 2016.

[23] N. R. Vanaparthi, "The Roadmap to Mainframe Modernization: Bridging Legacy Systems with the Cloud," 2025.

[24] J. Riihimäki, "Automatic testing and delivery in web development," 2019.

[25] W. G. Town, "Integration of microcomputer and mainframe information systems," J. Chem. Inf. Comput. Sci., vol. 31, no. 2, pp. 176-180, 1991.