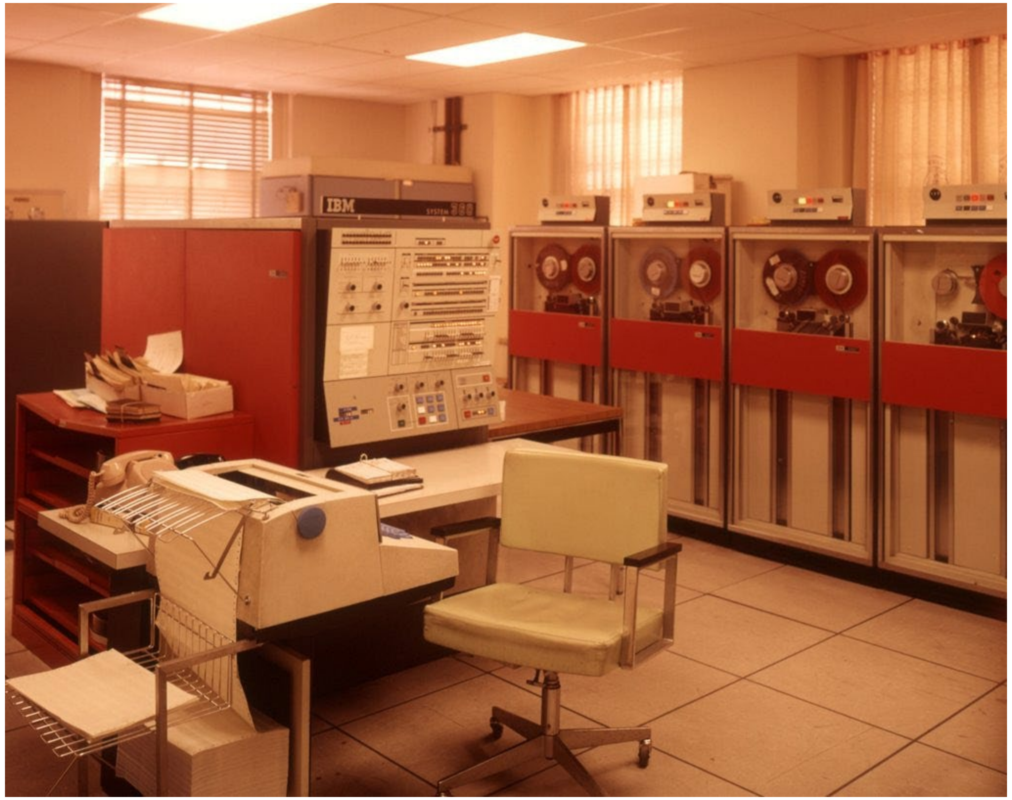
RISK AND RESILIENCE BOOTCAMP


TEKsystems
Global Services
WORKFORCE
DEVELOPMENT

LEGACY SYSTEMS

This section is an introduction to legacy systems

- What they are
- The IT challenge of legacy systems
- The risk and resilience issues associated with legacy systems



DEFINING LEGACY SYSTEM

- Systems, applications, or infrastructure that continue to be in use despite being outdated or replaced by newer technologies
- Characteristics
  - Often run on old hardware or software that is no longer supported by vendors
  - Difficult to modify or integrate with modern infrastructure
  - Critical to operations; the business can’t function without it
- Why they persist
  - Cost, complexity, business dependency, data lock-in, lack of replacement skills
  - They are often very complex and tightly integrated into IT infrastructure
  - Migration to a new system is often prohibitive because of the cost and size of the project
    - Often requires legacy technical skills that may not currently exist in the organization

LEGACY SYSTEM GENERATIONS

- There have been a number of “generations” of systems development
  - For example
    - Mainframes from the 1960-1980s
    - Object oriented web based systems from the 1990s-2010s
    - Big data and streaming systems 2010s-current
- An organization often has a mix of legacy systems
  - From different generations of development
- A legacy era has its own characteristics
  - Principles for designing systems architecture
  - Programming tools, languages and best practices for program design
  - Models for interacting with data and other systems
  - Development, testing and deployment methodologies

ERAS OF LEGACY SYSTEMS

Era	Technology Examples	Common Characteristics	Typical Industries
1960s–1970s: Mainframe Era	IBM System/360, DEC PDP, COBOL, JCL	Batch processing, centralized computing, card input/output	Banking, insurance, government
1980s: Minicomputers & Early Networks	VAX/VMS, HP 3000, UNIX systems	Departmental systems, basic networking	Manufacturing, education
1990s: Client–Server & Early ERP	Oracle DBs, SAP R/3, Lotus Notes	PC front-end with database back-end, Windows NT	Finance, retail, logistics
2000s: Web Applications & Custom Platforms	Early Java EE, .NET 1.x, Flash apps	Custom web portals, monolithic architecture	E-commerce, telecom
2010s: Early Cloud & Virtualization Legacy	VMware vSphere, AWS EC2 classic, private clouds	Aging virtualized infrastructure, proprietary APIs	Enterprise IT, healthcare

LEGACY SYSTEMS IN USE

- Mainframe-based systems
  - For example: COBOL based banking systems, airline reservation systems
- Outdated databases
  - For example: Oracle 8i, Sybase, DB2 z/OS, archaic network and hierarchical databases
- Monolithic applications
  - For example: Large enterprise resource planning or customer relationship management systems that are monolithic and resist modularization
- Custom-built software with no documentation
  - For example: Contracted software delivered with a support agreement instead of documentation and source code, common practice for projects in 1960s-1990s
  - For example: In house software that was never properly documented

LEGACY SYSTEMS IN USE

- Unsupported hardware
  - For example: proprietary industrial controllers, device drivers
- Older operating systems
  - For example: Windows XP, AIX 5L, Solaris
- Legacy network protocols
  - For example: SNMPv1, Telnet, FTP

LEGACY SYSTEM LONGEVITY

- High replacement cost
  - Replacing a legacy system can incur high costs along many dimensions
    - For example: hardware, software, migration, and retraining costs
  - ROI may not be immediate, especially if the legacy system still works
  - Common to defer modernization because budgets favor short-term operational spending over large transformation projects
- Example
  - A large insurance company may still use a COBOL-based policy system because the cost of rewriting and testing all its business rules is prohibitive

LEGACY SYSTEM LONGEVITY

- Mission critical dependence
  - Often perform core mission critical business functions
  - Deeply embedded in daily operations
  - Even brief outages can cause massive disruption
  - The mindset is often, ‘If it isn’t broken, don’t fix it.’
- Example
  - Airlines and banks rely on decades-old mainframes that process millions of daily transactions reliably

LEGACY SYSTEM LONGEVITY

- Reliability and stability
  - Legacy systems are reliable; they often have run for decades without major failures
  - Known performance under load and mature processes help with operations planning
  - Replacing them can introduce instability.
- Example
  - COBOL batch jobs running since the 1980s still process overnight settlements successfully

LEGACY SYSTEM LONGEVITY

- Data lock-in and integration complexity
  - Critical historical or regulatory data may reside in legacy databases
    - Data may be in formats, data models, or implementations that are hard to migrate safely
  - Modern systems might not be compatible with older data structures
  - Integrating or converting decades of data can risk data loss or corruption
- Example
  - Government systems holding citizen records in mainframe files that can’t easily be ported to relational databases

LEGACY SYSTEM LONGEVITY

- Business process entrenchment
  - Often tightly intertwined with business processes, policies, and staff routines
  - Replacement may require business re-engineering, retraining, and workflow redesign, not just new IT systems
  - Organizational inertia resists these changes because of perceived costs, potential down-time and loss of productivity during the transition
- Example
  - A logistics firm’s custom-built dispatch application mirrors decades of operational practices that no modern alternative can duplicate

LEGACY SYSTEM LONGEVITY

- Lack of suitable replacements
  - There may be no modern equivalent that can fully replace legacy functionality or compliance requirements
  - Custom logic built over years can't easily be reproduced in off-the-shelf systems
  - The costs of rewriting custom logic may be prohibitive
- Example
  - Nuclear facility monitoring systems or legacy SCADA software that control specialized equipment

LEGACY SYSTEM LONGEVITY

- Compatibility dependencies
  - Other systems often depend on legacy interfaces, file formats, or protocols
  - Removing one legacy component might break dozens of connected systems or scripts
- Example
  - Old ERP modules still in use because other downstream systems rely on their flat-file outputs

LEGACY SYSTEM LONGEVITY

- Vendor lock-in
  - Proprietary platforms can make it expensive or technically difficult to migrate away
  - Licensing and support contracts may force staying with legacy solutions rather than replacing
- Example
  - Organizations tied to specific IBM or Oracle mainframe environments

LEGACY SYSTEM LONGEVITY

- Skills availability
  - Organizations have staff who are experienced with the legacy technology
    - This allows them to maintain it reliably
  - Converting to new technology would require staff with new skill sets
  - Downside: What happens when the staff with the legacy skills retire?
- Example
  - A financial institution retaining COBOL developers nearing retirement because the cost of retraining new staff on modernization tools is higher
  - Still have to deal with the loss of the programmers after they retire

LEGACY SYSTEM LONGEVITY

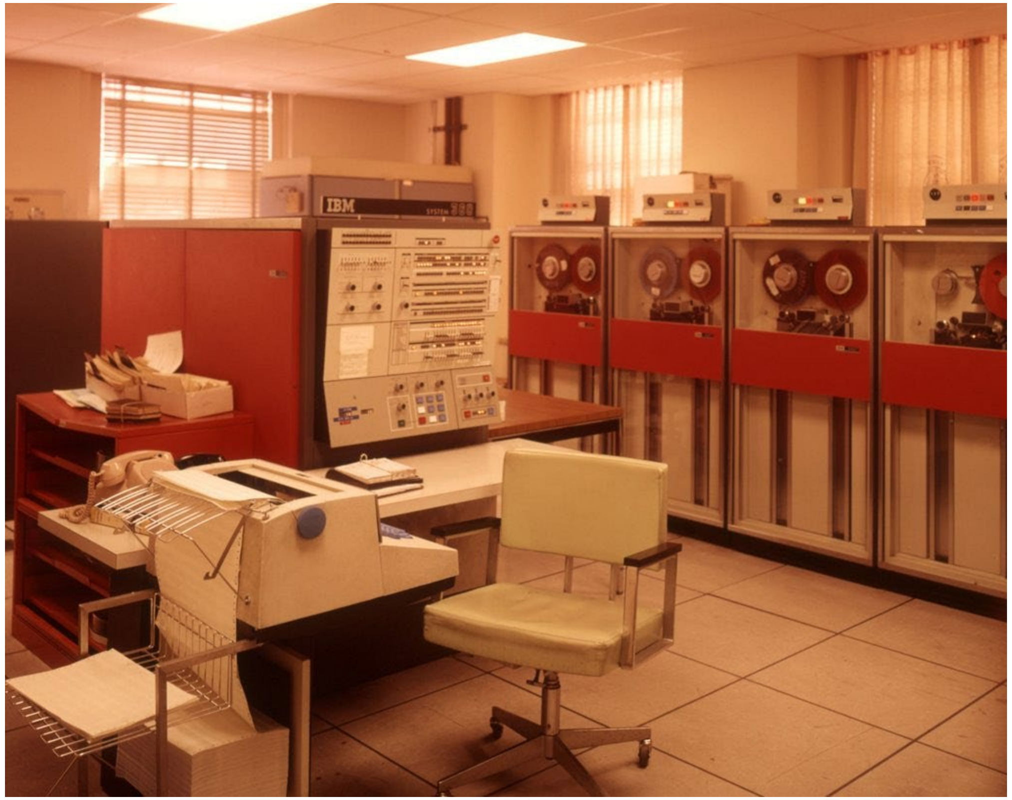
- Regulatory and compliance requirements
  - Legacy systems may be certified or approved for regulated use
    - For example: defense, healthcare, aviation or environment with high security requirements
  - Replacements would need to go through costly recertification or revalidation
  - Organizations keep the old system running within a controlled, compliant environment
- Example
  - Medical imaging systems that meet FDA approval under specific software versions

LEGACY SYSTEMS STILL IN USE

Banking

COBOL-based core banking still running on mainframes in large banks (e.g., JPMorgan Chase, Bank of America)

- Risk:
  - Shortage of COBOL developers, integration complexity, high operational risk
- Resilience factor:
  - Proven reliability, decades of uptime, transaction integrity

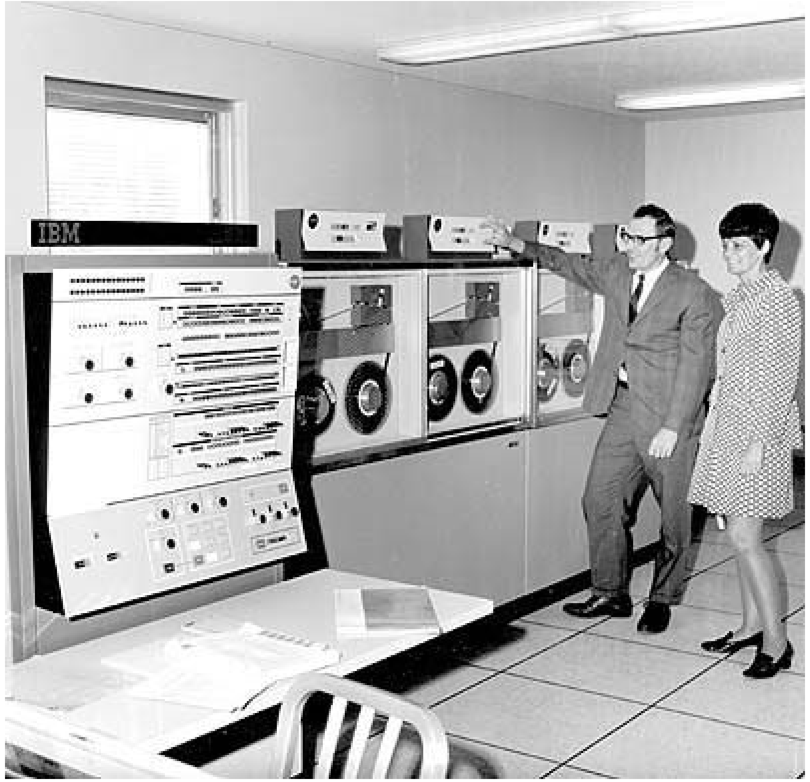


LEGACY SYSTEMS STILL IN USE

Government Systems:

U.S. IRS and Social Security still depend on 1970s-era mainframes.

- Risk:
  - High maintenance costs, cybersecurity gaps, data integration issues
- Resilience issue
  - Difficulty scaling for new service demands



LEGACY SYSTEMS STILL IN USE

Airline Industry:

SABRE reservation system (1950s origin, IBM mainframe roots)

- Risk:
  - Complex dependency chains; single-point failures have caused major outages
- Resilience issue:
  - Modernization attempts risk service disruption

LEGACY SYSTEMS STILL IN USE

Healthcare:

Hospital record systems using legacy Windows servers and outdated medical devices

- Risk:
  - Compliance exposure (HIPAA), vulnerability to ransomware
- Resilience issue:
  - Data migration difficulty; limited downtime windows

LEGACY SYSTEMS STILL IN USE

Industrial/OT Environments:

Power grids and manufacturing plants using SCADA systems built in the 1980s-90s

- Risk:
  - Insecure protocols, physical process disruption potential
- Resilience issue:
  - Patching can disrupt production; reliance on physical redundancy

LEGACY SYSTEM RISK

- Operational risk
  - The risk that legacy systems fail to operate as intended due to aging components, obsolete dependencies, or lack of available spare parts
  - How it can occur:
    - Hardware failure in outdated servers, storage arrays, or networking gear that is no longer manufactured
    - Software instability due to untested patches or operating systems no longer maintained
    - Downtime caused by failure in components that can't easily be replaced or replicated
    - Inability to restore operations quickly after an incident due to lack of recovery media or compatible hardware
  - Example:
    - A government tax agency running a 1980s mainframe suffers a hardware failure; replacement parts are only available through eBay or custom refurbishers, leading to weeks of downtime
  - Resilience Impact:
    - Low adaptability and recovery speed
    - Systems may be stable in day-to-day use but highly brittle under stress, creating single points of failure

LEGACY SYSTEM RISK

- Cyber risk
  - The risk of security compromise stemming from unpatched vulnerabilities, outdated cryptographic protocols, or unsupported software components
  - How it can occur
    - Inability to apply modern security patches because the OS or application is out of support
    - Use of insecure network services (e.g., Telnet, FTP, SMBv1)
    - Lack of encryption or modern authentication mechanisms
    - Older firmware that can’t be monitored by current intrusion detection systems
    - Legacy systems serving as “soft targets” or entry points for attackers
  - Example:
    - During the 2017 WannaCry ransomware outbreak, many healthcare systems were compromised because they still ran Windows XP, which was no longer supported or patched at the time
  - Resilience Impact:
    - Weakens the overall security posture of the organization and increases incident likelihood. Once compromised, legacy systems are often hard to recover or rebuild, compounding the damage

LEGACY SYSTEM RISK

- Compliance risk
  - The risk of violating current legal, regulatory, or industry requirements due to the legacy system's inability to meet modern compliance standards
  - How it can occur:
    - Failure to comply with data protection laws (e.g., GDPR, HIPAA) due to lack of encryption, audit trails, or access controls
    - Inability to produce required records or logs during audits
    - Use of outdated algorithms or storage mechanisms that violate current standards
    - Certification or accreditation (e.g., PCI DSS) lost due to unsupported software or unverified security controls
  - Example:
    - A regional bank's transaction processing system stores customer data in plaintext files. Under new data privacy regulations, this storage method violates encryption requirements
  - Resilience Impact:
    - Legal exposure, financial penalties, and reputational damage erode organizational resilience
    - Maintaining compliance requires either compensating controls or accelerated modernization

LEGACY SYSTEM RISK

- Vendor dependency (lock-in risk)
  - The risk arising from overreliance on a single vendor or proprietary technology that cannot be easily replaced or supported by others.
  - How it can occur:
    - Vendor stops providing updates or support (end-of-life declaration)
    - Escalating maintenance costs due to exclusive vendor control
    - Inability to migrate data or interfaces because of proprietary formats
    - Delays in incident resolution when vendor assistance is unavailable or slow
  - Example:
    - A manufacturing company relies on an industrial control system that only one vendor can service
    - When the vendor is acquired and discontinues the product, the company faces unplanned obsolescence
  - Resilience Impact:
    - Creates single points of dependency that undermine operational flexibility
    - If the vendor collapses or changes terms, the organization’s critical operations are directly exposed

LEGACY SYSTEM RISK

- Integration Risk
  - The risk that legacy systems cannot effectively interface with new technologies, cloud platforms, or data ecosystems, creating operational silos or broken workflows
  - How it can occur:
    - Legacy applications that lack APIs or modern interface standards
    - Incompatibility with cloud migration efforts (e.g., no containerization support)
    - Manual data transfer between systems leading to errors and delays
    - High complexity in integration projects leading to scheduling or cost overruns
    - "Shadow IT" workarounds to bridge systems informally, introducing new risks
  - Example:
    - A logistics company attempts to link its 1990s warehouse management software to a new SaaS analytics dashboard but must rely on nightly flat-file exports due to lack of API support
  - Resilience Impact:
    - Reduces organizational agility and visibility, hindering coordinated responses during incidents
    - Inability to integrate with newer monitoring, backup, or automation tools weakens resilience

LEGACY SYSTEM RISK

- Skill risk (knowledge and workforce risk)
  - The risk of operational failure or business disruption caused by the loss of personnel who understand and can maintain the legacy system.
  - How it can occur:
    - Retirement or departure of long-time staff who built or maintained the system
    - Inadequate documentation or institutional memory
    - Difficulty finding new talent trained in obsolete technologies (e.g., COBOL, PowerBuilder, VAX/VMS)
    - Over-reliance on a few experts (“key-person risk”)
  - Example:
    - A telecom company’s billing platform runs on an aging UNIX variant
    - Only two engineers know its configuration; one retires and the other moves on, leaving no internal capability for updates
  - Resilience Impact:
    - Severely limits the organization’s ability to adapt, recover, or troubleshoot failures
    - Skills loss magnifies every other risk category: operational, cyber, and compliance

LEGACY SYSTEM RISK

Risk Type	Primary Threats	Resilience Impact
Operational	Hardware/software failure, downtime	Reduced recoverability, brittle operations
Cyber	Unpatched vulnerabilities, obsolete security	Increased attack surface, weak incident recovery
Compliance	Regulatory nonconformance, audit failure	Legal penalties, reputational damage
Vendor Dependency	End-of-life support, proprietary lock-in	Limited flexibility, slow recovery
Skill	Retirement, undocumented systems	Knowledge loss, higher MTTR (Mean Time to Repair)
Integration	Incompatibility, data silos	Slower modernization, weak systemic resilience

LEGACY PHASE OUT

Approach	Description	Benefits	Challenges
Rehosting ("Lift and Shift")	Move the legacy application to new infrastructure (e.g., cloud or virtualized platform) with minimal code change.	Quick to execute, immediate hardware risk reduction.	Doesn't address outdated code or architecture.
Replatforming ("Lift, Tinker, and Shift")	Migrate to a new platform or runtime with limited optimization (e.g., from AIX to Linux, or on-prem Oracle to AWS RDS).	Balances cost and modernization; improved performance and security.	Still retains legacy design constraints.
Refactoring / Re-architecting	Rewrite or restructure the code to improve maintainability and integrate with modern technologies.	Extends system lifespan, enables microservices or APIs.	Expensive and time-consuming; risk of disruption.
Rebuilding	Recreate the system using modern technologies and architectures, preserving business logic but replacing code entirely.	Enables long-term agility, compliance, and resilience.	Highest cost and complexity; long transition period.
Replacement	Replace the system with a commercial off-the-shelf (COTS) or SaaS alternative.	Eliminates legacy dependencies.	Requires major business process change and retraining.
Encapsulation	Wrap legacy systems with APIs or middleware to expose functionality without altering the system.	Allows integration with modern applications and cloud services.	Risk and performance issues may persist inside the legacy core.
Retirement	Gradually decommission redundant legacy systems after data migration or business process redesign.	Reduces maintenance burden.	Risk of data loss or incomplete migration if not carefully managed.

RISK MITIGATION

- Security hardening
  - Network segmentation
    - Isolate legacy systems from the internet and modern networks
  - Access control and monitoring
    - Implement firewalls, multi-factor authentication, and strict logging
  - Virtual patching
    - Use intrusion prevention systems to block exploits when patches can't be applied
  - Application firewalls or proxies:
    - Filter unsafe traffic to or from legacy interfaces
  - Example
    - A bank maintains a COBOL mainframe but isolates it on a restricted subnet and monitors access through a SIEM (Security Information and Event Management) platform

RISK MITIGATION

- Data protection and backups
  - Implement modern backup solutions even for legacy environments
  - Store critical data in interoperable, exportable formats (CSV, XML, or JSON)
  - Introduce regular recovery drills to test recovery speed and completeness
  - Resilience Effect:
    - Improves recovery capability and ensures data continuity in case of failure or ransomware attack
- Documentation and knowledge preservation
  - Capture institutional knowledge before key staff retire
  - Document workflows, configurations, and dependencies
  - Use knowledge management tools and automated documentation extractors
  - Resilience Effect
    - Prevents “key-person risk” and ensures continuity when maintaining or migrating systems

RISK MITIGATION

- Virtualization and emulation
  - Run legacy applications on virtual machines or emulators that replicate old hardware environments
  - Preserves system functionality while allowing modern management tools and redundancy
  - Example
    - A manufacturing firm runs a 1990s DOS-based control system in a virtualized sandbox on modern hardware, ensuring continued support without physical dependence on aging equipment
- Controlled modernization (“Strangler Pattern”)
  - Introduce new modules alongside the legacy system
  - Gradually replace or bypass old functions without a “big bang” rewrite
  - Common in large-scale enterprise or government modernization programs
  - Resilience Effect
    - Balances risk by maintaining service continuity while reducing long-term dependency

RISK MITIGATION

- Skills and vendor support
  - Partner with specialized vendors or consultants who still support legacy technologies
  - Create cross-training programs for younger staff.
  - Maintain small internal “legacy stewardship teams” to ensure stability during transition
  - Resilience Effect:
    - Maintains operational knowledge and minimizes disruption risk during modernization

RISK STRATEGY

- Governance and risk planning
  - Establish a legacy modernization roadmap within the organization’s IT risk management framework
  - Prioritize systems based on criticality, risk exposure, and business impact
  - Integrate modernization goals into annual IT risk and resilience objectives
- Dual-environment operation
  - Operate old and new systems in parallel for a defined transition period
  - Enables validation, testing, and fallback capability

RISK STRATEGY

- Change management and testing
  - Introduce automated regression testing for legacy systems before any migration step
  - Implement continuous monitoring and fallback plans
- Compliance Alignment
  - Map modernization projects to regulatory requirements
  - Especially data protection and audit trail retention.
  - Ensure that modernization itself doesn’t create new compliance risks

RISK STRATEGY

Objective	Approach	Resilience Benefit
Reduce operational fragility	Virtualization, cloud rehosting	Improved redundancy and recoverability
Improve security posture	Segmentation, virtual patching, access control	Reduced cyber exposure
Retain institutional knowledge	Documentation, cross-training	Sustained support capability
Prepare for long-term transformation	Strangler pattern, incremental modernization	Controlled risk during transition
Strengthen governance	Roadmaps, risk-based prioritization	Aligned modernization with resilience goals

Q&A AND OPEN DISCUSSION