Journal of Information Systems Engineering and Management
2025, 10(63s)
e-ISSN: 2468-4376
https://jisem-journal.com/

Research Article

Modernizing Legacy Healthcare Systems: An API-First, Event-Driven Architecture Approach

Neeraj Gaddam
Independent Researcher, USA

ARTICLE INFO

Received: 06 Nov 2025

Revised: 20 Dec 2025

Accepted: 29 Dec 2025

ABSTRACT

Contemporary healthcare institutions confront escalating demands to modernize outdated systems while preserving operational continuity and regulatory adherence. This article investigates architectural strategies for transforming legacy healthcare platforms through interface-prioritized, event-oriented methodologies. It examines financial and clinical obstacles created by aging healthcare infrastructure while introducing theoretical frameworks guiding effective monolith decomposition. The botanical replacement pattern receives particular attention as an implementation strategy specifically tailored for healthcare environments, highlighting a gradual transformation that sustains operational stability. Interface-driven connectivity emerges as a governance structure featuring a layered architecture of foundational, process-orchestration, and experience-delivery interfaces enabling healthcare enterprises to harmonize compliance requirements with innovation capabilities. Through strategic application of these architectural patterns, healthcare organizations successfully navigate complex transitions from unified legacy systems to adaptable, modular architectures while preserving patient safety and regulatory conformity throughout transformation initiatives.

Keywords: Legacy Platform Transformation, Interface-Prioritized Architecture, Incremental Replacement Pattern, Domain-Oriented Design, Healthcare Data Exchange

I. Introduction and Strategic Imperatives

Today's healthcare systems are under tremendous pressure to replace aging systems. They must do this while continuing to deliver service and meeting existing regulatory obligations. Many legacy technology systems have been in use for decades. They consume resources and inhibit clinical work at a time when digital transformation is critical. It will explore what is driving healthcare organizations to modernize and an architecture leveraging API and event-driven design.

Money matters first. Maintaining old healthcare systems creates real barriers to innovation. Industry experts point out that IT budgets get swallowed up by keeping legacy systems running. This leaves little funding for forward-thinking projects. The financial burden comes from several directions: old programming languages that few people know, inflexible licensing that doesn't match organizational needs, and growing hardware costs for maintaining on-site infrastructure designed for peak usage rather than average needs. There's also the hidden cost of delayed innovation and reduced market responsiveness, which puts additional strain on healthcare organizations already operating with thin profit margins in competitive markets.

But it goes beyond money. Old systems directly hurt clinical effectiveness and patient care. Research shows that healthcare providers waste hours completing non-clinical work while relying on legacy applications. Their workflows are hampered by cluttered user interfaces, attempting to construct their work from disparate information systems. The modern-day healthcare platforms came with user interfaces that were designed and built before any modern usability requirements. They require excessive clicking, compel users to pull information from disparate and disconnected screens, and lack

Journal of Information Systems Engineering and Management
2025, 10(63s)
e-ISSN: 2468-4376
https://jisem-journal.com/

Research Article

workflows that mirror what clinicians do. These concerns compel healthcare workers to create workarounds, duplicate records across multiple systems, and manually reconcile and correct conflicting entries. All this takes away from patient care time while increasing the risk of medical errors. The usability issues also contribute significantly to burnout among healthcare professionals while reducing time spent with patients - a concerning issue in resource-limited healthcare environments.

The necessity for modernization is only amplified when assessing the state of healthcare technology across America. A comprehensive survey of hospital physiology demonstrates that a number of U.S. healthcare organizations still manage a significant portion of care-related functions on antiquated platforms that are fundamentally incompatible with the capabilities and efficiencies offered by cloud technology, modern APIs, or interoperability standards. These systems were conceptualized in a time when the healthcare industry was relatively and primarily focused exclusively on facilities - as opposed to patients; thereby demonstrating a fundamental incompatibility when considering the models and methods of care employed today, which require coordination across settings, telehealth, remote monitoring, and engagement with the patient. Legacy systems usually utilize old database technologies, proprietary communication protocols, and monolithic architectures that inherently make them resistant to integration into modern digital health networks, which explicitly limits the potential of the healthcare organization to reap the benefits of utilizing these emerging technologies (e.g., artificial intelligence; predictive analytics, remote-monitoring solutions, etc.) in ways that could yield significant enhancements in clinical outcomes and operational efficiencies.

Healthcare has a unique threshold of challenges relating to modernization due to the numerous regulatory requirements regarding the use, availability, and compliance reporting of patient data. Unlike other industries, healthcare organizations must operate within the full construct of understanding the ethical landscape and operationalize compliance with respect to HIPAA, interoperability standards as enabled by FHIR, and mandated security requirements as a part of updates to their systems. The regulatory landscape creates unique technical constraints around data encryption, audit trails, patient consent management, minimum necessary data access, and vendor agreements - all factors requiring careful consideration during modernization efforts. Due to the impact on patient care, healthcare systems must also maintain extremely high availability, making any alternative migration strategies involving downtime far more difficult. These regulatory and operational requirements suggest the need for focused modernization efforts that ensure compliant systems while leveraging technical innovation.

According to this article, an architectural approach they have identified as API-first, event-driven architecture can serve as a workable approach to modernizing legacy technology within a healthcare organization. By employing the Strangler Fig pattern within an API-led connectivity model, organizations can incrementally refactor monolithic applications while also ensuring that existing operations remain stable and regulated. The transformation process begins with building a facade layer that captures requests to the monolithic system and allows the organization to redirect requests from that layer to new and refactored services, depending on the functionality being performed. The facade layer will make it so that requests processed by the monolithic system remain operational and regulated during transition. The API-first mentality presumes that interfaces are first-order artifacts, so API contracts are designed to exist before implementation and remain unchanged as guaranteed quality contracts between services, sterilized from the technology used to develop either service. By adapting to this approach, a healthcare organization can substitute monolithic components incrementally but still maintain the user's functional perspective that will appear unchanged. Subsequently, the event-driven communication paradigm also affords developers to achieve loose coupling between services, which is advantageous to improving the resiliency and scalability of the platform. The transition to loose coupling is also valuable because it allows for a shift from consuming behavioral events with synchronous message patterns to sharing information as behavioral events

Journal of Information Systems Engineering and Management
2025, 10(63s)
e-ISSN: 2468-4376
https://jisem-journal.com/

Research Article

between distributed applications in asynchronous behavioral event processing models during development that are aligned with the distributed processing patterns in modern healthcare.

II. Theoretical Foundations for Monolith Decomposition

Moving from big, all-in-one systems to smaller, independent services marks a big shift in how it design healthcare systems are designed during digital transformation. This section covers the key ideas that guide the successful breaking down of monoliths, focusing on how they work in heavily regulated healthcare settings.

The shift from monoliths to microservices has happened because enterprise apps keep getting more complex, and organizations need to move faster. Traditional healthcare systems with tightly connected parts sharing one codebase and database make innovation hard because you have to develop, test, and release the whole thing at once. Monoliths are adequate for small applications at first, but introduce challenges as the systems grow. Technical debt accumulates as systems evolve and development becomes more complicated, slower, and more risky to change. Healthcare organizations with monoliths often experience similar challenges. In particular, there are bottlenecks where even small changes require a redeployment, are locked into technologies that prohibit consideration of newer frameworks, and must scale the entire application even when certain functions may just be throttled. Team coordination is sometimes challenging, too, since a large development team can only work around a single codebase. These challenges can take a toll within the healthcare space when quick responses to regulations, consideration of new technologies, and adjusted care models to stay competitive in the landscape are viewed as paramount.

Domain-Driven Design gives healthcare organizations a sophisticated method for breaking down monolithic applications by aligning technical design with underlying business areas. The Strangler Fig approach, a key method within this framework, gets its name from how fig vines gradually cover host trees, eventually replacing them while keeping the same basic structure. In software modernization, this pattern creates a facade or API layer that catches calls to the old system and gradually redirects them to new microservice implementations. This lets healthcare organizations migrate functionality bit by bit without risky "all at once" replacements that could cause operational problems. Implementation typically starts by identifying system boundaries, creating a facade layer between clients and the legacy system, and then gradually building replacement services behind this facade. As new services prove themselves, the facade sends more traffic to them until the old component can be safely retired. This careful approach works particularly well in healthcare, where system availability directly affects patient care and where thorough testing is needed to ensure clinical safety and regulatory compliance before switching functions over.

The DDD concept of bounded contexts helps manage the complexity in healthcare domains where terminology and processes vary across departments and specialties. Strangler Pattern implementation specifically uses these bounded context ideas to find logical boundaries for breaking things up. This follows a structured approach starting with a thorough analysis of the existing monolith to identify function clusters with minimal dependencies between them. After finding these boundaries, teams set up an interception layer—typically an API Gateway—that handles communication between clients and backend services. This gateway initially sends most requests to the legacy monolith while gradually adding routing rules that direct specific functionality to newly developed microservices. The pattern focuses on keeping both implementations alive and well during transitional times, using a validation period to ensure that both implementations operate consistently. In healthcare systems, this validation period is important to ensure that patient data is still valid, that regulatory compliance is being satisfied, and that the clinical decision support remains in place for continuity. Gradual patterns allow organizations the freedom to prioritize decomposition of their systems based on business value, technical debt, or other variables associated with risk—enabling organizations to jump straight into

Journal of Information Systems Engineering and Management
2025, 10(63s)
e-ISSN: 2468-4376
https://jisem-journal.com/

Research Article

modernizing the areas of their system they have deemed most troublesome, while also being conscientious of the other aspects of the system.

DDD Concept	Healthcare Application	Modernization Benefit
Bounded Contexts	Separation between clinical, administrative, and financial domains with distinct terminologies	Provides natural boundaries for microservice decomposition while respecting domain-specific language [6]
Ubiquitous Language	Shared vocabulary between technical teams and healthcare domain experts	Reduces translation errors in requirements, particularly critical for clinical terminology implementation
Aggregates and Entities	Clinical objects like patient records, orders, and results with clear ownership boundaries	Establishes data consistency rules appropriate for each healthcare domain

Table 2: Domain-Driven Design Elements in Healthcare Modernization.[6]

Data ownership and consistency create special challenges in regulated healthcare environments where information accuracy directly impacts patient safety and regulatory compliance. Real-world experiences from major enterprises show that database decomposition is often the biggest challenge in monolith migration projects. Leading organizations have addressed these challenges through incremental approaches that start by identifying data domains aligned with bounded contexts, then implementing read-only copies before moving to fully independent data stores. Many successful healthcare modernization projects use a hybrid approach where transactions initially stay within the legacy database while gradually transitioning to distributed data patterns as microservices mature. These organizations have found that implementing Command Query Responsibility Segregation provides a useful intermediate step, allowing read operations to move to microservices while keeping write operations in the legacy system until data migration is fully verified. Implementing eventual consistency models requires special care in healthcare environments where certain data relationships—like medication allergies, current prescriptions, and diagnostic results—have direct patient safety implications that may require stronger consistency guarantees than other domains. Successful implementations have addressed these concerns through careful domain analysis that distinguishes between data requiring immediate consistency versus areas where eventual consistency provides acceptable tradeoffs between system coupling and data accuracy.

Successful healthcare modernization endeavors exhibit a balance of strategic and tactical phases of domain analysis such that decomposition decisions meet immediate organizational needs and align with longer-term organizational needs. In the strategic phase, techniques such as Event Storming—where a group of domain experts and technical team members work together to map business processes with a collection of sticky notes with domain events, commands, and aggregates—are employed. These activities uncover natural system boundaries by displaying how information moves from one team or individual to another, demonstrating an organization's transaction patterns and responsibilities. The tactical phase implements specific patterns—such as the Strangler Fig—through technical activities sequenced in certain ways, such as creating facade interfaces, creating traffic routing mechanisms, creating new microservices, and gradually migrating the actual state of functionality. Many healthcare organizations have found it particularly effective to use clinical and administrative workflows as a means to decompose the system instead of technical bounds, which permits microservices aligned with a coherent business process recognized by domain experts. This alignment helps technical teams collaborate better with healthcare professionals during the design and testing phases, while also ensuring that redesigned workflows reflect the clinical levels of reality. Healthcare modernization projects further benefit from emphasizing the "seam" concept—identifying natural boundaries in the monolith where functionalities can be separated with minimal disruption to interrelated components. These seams often align with different rates of change, separate regulatory

Journal of Information Systems Engineering and Management
2025, 10(63s)
e-ISSN: 2468-4376
https://jisem-journal.com/

Research Article

domains, or distinct user groups, providing natural decomposition boundaries that minimize refactoring requirements while maximizing business value delivery.

III. The Strangler Fig Pattern: Implementation in Healthcare Contexts

The Strangler Fig approach has become crucial for healthcare groups updating legacy systems while maintaining continuous operations [9]. Much like its botanical namesake, which wraps itself around a host tree until it has enveloped it completely, this approach provides a systematic way to transition from monolithic applications to modern microservice architectures. This section details implementation considerations unique to healthcare environments, where regulatory obligations and requirements related to the safety of patients create additional complexity to modernizing efforts.

Healthcare modernization particularly benefits from the Strangler Fig pattern's emphasis on incremental transformation rather than high-risk complete replacements [9]. Underlying this pattern is an intermediary layer—usually realized through an API gateway—that serves as the middleman between the clients and the legacy system, allowing incremental redirection of traffic away from the old components to newly built microservices. Its realization in healthcare environments adds Domain-Oriented Microservice Architecture principles in organizing services around specific business capabilities, not technical functions. This domain-based methodology directly maps to the complex, multifaceted nature of healthcare's organizational structure, where patient registration, clinical documentation, medication management, revenue cycle, and population health analytics all have distinct business rules, operational processes, and compliance mandates. Healthcare organizations generally start with collective domain discovery sessions where clinical professionals, operational leaders, and technical experts together chart the organizational domain territory, including natural boundaries that correspond to integrated business functions with few cross-domain dependencies [9]. These boundaries then determine microservice responsibility segmentation, where every service owns sole responsibility for specific business capabilities and corresponding data assets. Healthcare-specific extensions include tailored governance frameworks providing regulatory compliance throughout transformation, specifically addressing HIPAA requirements, clinical documentation standards, and standards of interoperability like FHIR. Implementation introduces conventional approaches with robust audit capability, providing transaction traceability across distributed services, tailored validation steps for clinical decision support rules, and sophisticated monitoring capabilities that detect potential variances before impacting patient care delivery.

The Strangler Fig implementation follows a structured three-phase methodology providing healthcare organizations with a clear transformation roadmap: Transform, Coexist, and Eliminate [10]. The initial Transform phase encompasses establishing the interception facade, implementing preliminary routing mechanisms, and developing initial replacement services—typically focusing on relatively independent functions with minimal integration complexity. This phase leverages Domain-Driven Design's strategic principles to identify bounded contexts—coherent functional areas with clearly defined boundaries and explicit interfaces to adjacent domains. For healthcare organizations, identifying bounded contexts needs particular emphasis, where the meaning attached to terms can differ by clinical specialization, administration of departments, or regulatory requirements. For example, concepts such as "patient admission" have different meanings, workflows, and data required by emergency care, inpatient care, or ambulatory care. The Transform phase employs specialized modeling techniques such as Context Mapping to document relationships between bounded contexts, implementing integration patterns including Conformist, Customer-Supplier, and Anti-Corruption Layer [10]. The Anti-Corruption Layer proves especially valuable in healthcare modernization, providing translation mechanisms between legacy systems and new microservices that prevent outdated data models and business rules from propagating into modernized architecture. At this stage, healthcare organizations develop a well-defined ubiquitous language for each bounded context,

Journal of Information Systems Engineering and Management
2025, 10(63s)
e-ISSN: 2468-4376
https://jisem-journal.com/

Research Article

consistently documenting terminology used by technical specialists as well as by domain expert. The shared terms help to mitigate the translation errors, often seen and troublesome in healthcare IT implementations, where the misinterpretation has resulted in usability issues for users, alterations to their workflow, and ultimately the potential harm to patients.

Risk mitigation of the Strangler Fig is especially highlighted and advantageous in healthcare environments where system failure could hurt patient outcomes. [11] Through incremental component replacement, organizations limit change scope, enabling comprehensive testing and simplified rollback capabilities when issues emerge. The pattern implements risk reduction through a systematic four-step migration methodology addressing both technical and domain challenges throughout transformation. The process begins with comprehensive business domain analysis, where organizations develop a thorough understanding of operational landscapes, identifying core domains providing competitive differentiation, supporting domains enabling essential operations, and generic domains representing common industry functionality. This analysis guides prioritization decisions, directing modernization efforts toward high-value domains while potentially leveraging commercial solutions for generic functions. The second step involves defining bounded contexts and establishing clear ownership boundaries for data and business rules, creating conceptual decomposition frameworks aligned with organizational structures rather than technical implementations [11]. The third step applies tactical DDD patterns, including aggregates, entities, value objects, and domain events, to model bounded context internals, creating implementation blueprints accurately reflecting domain requirements and constraints. The final step identifies candidate microservices based on domain analysis, ensuring each service maintains clear, cohesive responsibility rather than arbitrary technical divisions. This domain-driven approach substantially reduces several risk categories in healthcare modernization: it mitigates technical risk by ensuring service boundaries align with natural business domain segmentation; it reduces organizational risk by creating services that map directly to business capabilities teams readily comprehend; and it decreases operational risk by limiting change scope to well-defined domain areas with clear boundaries and explicit integration points.

Maintaining operational continuity during transformation presents particular challenges in healthcare environments where continuous availability directly impacts patient care [12]. Successful implementations employ multiple strategies addressing this challenge, including scheduling significant transition activities during reduced clinical activity periods, implementing sophisticated traffic management capabilities enabling immediate request redirection when issues emerge with new components, and establishing enhanced support models throughout transition periods. Real-world migration experiences highlight several critical lessons healthcare organizations should incorporate into modernization strategies to maintain continuity throughout transformation journeys. Paramount among these is establishing comprehensive test automation before commencing refactoring, creating protective mechanisms quickly identifying unintended consequences from architectural modifications. Healthcare implementations require particularly robust testing strategies validating not merely technical functionality but additionally clinical appropriateness, regulatory compliance, and workflow efficiency across system boundaries [12]. Another essential lesson involves incrementally reducing monolith footprint by transforming legacy systems into routing proxies, where original applications eventually function primarily as orchestrators calling modernized microservices rather than directly implementing business logic. This approach maintains operational consistency from user perspectives while progressively transitioning functionality to new architectures. Healthcare organizations should avoid simultaneously adopting multiple unfamiliar technologies during modernization, instead focusing on established, proven patterns rather than emerging approaches that introduce additional risk factors. Successful implementations emphasize maintaining database consistency through carefully orchestrated data migration strategies, recognizing that database decomposition frequently represents the most challenging aspect of healthcare modernization, given

Journal of Information Systems Engineering and Management
2025, 10(63s)
e-ISSN: 2468-4376
https://jisem-journal.com/

Research Article

complex relationships between clinical, administrative, and financial data spanning multiple bounded contexts.

Implementation Phase	Key Activities	Healthcare-Specific Considerations
Transform	Establishing API facade and implementing initial routing mechanisms	Begin with non-clinical functions to demonstrate value with minimal patient safety risk [9]
Coexist	Gradual traffic shifting with legacy and new systems operating simultaneously	Implement comprehensive validation protocols and clinical safety monitoring during extended parallel operations
Eliminate	Decommissioning legacy components after complete functionality migration	Verify all regulatory compliance requirements are maintained in the new architecture before removal

Table 3: Strangler Fig Pattern Implementation Phases in Healthcare. [9]

Healthcare organizations have successfully implemented the Strangler Fig pattern across diverse enterprise contexts, demonstrating effectiveness in complex, regulated environments [12]. In-depth examinations of actual implementations demonstrate seven key success factors healthcare organizations need to include in modernization plans. First, thorough test automation should come before refactoring, building preventive mechanisms to catch regressions and side effects before they affect operations. Second, organizations can gradually rework monoliths into reverse proxies forwarding requests to new microservices instead of total replacement, ensuring operational continuity during changeover. Third, successful implementations are centered on refactoring strategies instead of whole rewrites, staying true to institutional knowledge locked in legacy code while incrementally enhancing structure and maintainability. Fourth, organizations need to plan strategically in upgrading efforts, focusing first on infrastructure enhancement before refactoring applications to set firm transformation roots [12]. Fifth, healthcare modernization benefits from pattern-based approaches rather than novel architectures, leveraging established practices demonstrating success in similar environments. Sixth, database decomposition requires particular attention, with healthcare organizations developing specialized strategies to maintain data integrity across bounded contexts while progressively transitioning toward distributed data models. Lastly, effective implementations accept that some legacy components will persist forever, targeting high-value areas of modernization and making pragmatic accommodations for full replacement to be impossible or economically unsustainable in some situations. These best practices from effective implementations offer practical advice for healthcare organizations undertaking modernization initiatives, providing tested solutions to overcome challenging issues, remodeling crucial healthcare systems while ensuring operational continuity.

IV. API-Led Connectivity: A Governance Framework for Healthcare Modernization

Modern healthcare enterprises encounter distinct obstacles when updating technological infrastructures while adhering to rigorous compliance mandates. API-led connectivity emerges as an effective governance framework addressing these dual imperatives, offering structured methodologies for system integration and modernization [13]. This segment examines how this architectural strategy, with differentiated API layers, empowers healthcare providers to balance stability, security, and innovation along their transformation journey.

API-led connectivity provides healthcare businesses with a three-tiered architectural framework drawing clear boundaries between system access, business process orchestration, and experience delivery. This stratified methodology creates governance structures where individual API tiers fulfill

Journal of Information Systems Engineering and Management
2025, 10(63s)
e-ISSN: 2468-4376
https://jisem-journal.com/

Research Article

specific functions with tailored design considerations and access controls. System APIs constitute the foundation layer, delivering secure, controlled access to essential healthcare information repositories, including electronic health records, laboratory information systems, diagnostic imaging archives, and administrative databases [13]. Implementing System APIs within healthcare contexts necessitates specialized infrastructure considerations that substantially affect performance, security, and operational reliability. Organizations pursuing modernization initiatives discover that container technologies coupled with orchestration platforms deliver fundamental capabilities for managing these mission-critical interfaces. The containerized methodology enables System APIs to undergo consistent development, testing, and deployment across environments, ensuring identical security controls and compliance mechanisms throughout the development-production continuum. Healthcare implementations typically establish geographically distributed deployments ensuring continuous availability, essential for maintaining uninterrupted access to vital patient information during regional service disruptions or scheduled maintenance activities. Infrastructure automation substantially contributes to maintaining security postures, with programmatic infrastructure management ensuring consistent configuration and security patches across all instances. Deployment architectures frequently incorporate dedicated database connection services enforcing additional security measures before backend system access, specialized intermediate storage mechanisms improving performance while maintaining stringent data currency requirements for clinical information, and comprehensive monitoring frameworks detecting abnormal access patterns potentially indicating security incidents [13]. This robust foundation enables System APIs to fulfill stringent availability requirements critical for healthcare operations while simultaneously supporting incremental modernization of backend systems without disrupting essential care delivery functions.

API Tier	Primary Function	Healthcare Implementation Focus
System APIs	Provide secure access to core healthcare data sources	Implement robust authentication, detailed audit logging, and HIPAA-compliant data access controls [13]
Process APIs	Orchestrate business logic and clinical workflows across domains	Enforce clinical protocols, regulatory requirements, and maintain state for long-running healthcare processes
Experience APIs	Deliver context-appropriate data to diverse healthcare applications	Tailor information presentation based on user roles, clinical context, and channel-specific requirements

Table 4: Three-Tiered API Architecture for Healthcare Modernization. [13]

Process APIs form the intermediate tier within the connectivity framework, coordinating business logic and workflow orchestration across functional domains within healthcare enterprises. These APIs aggregate and transform data from multiple System APIs to implement comprehensive business processes, including patient registration, medication management, or claims processing [14]. Transitioning monolithic healthcare systems toward domain-oriented microservices presents specific challenges and opportunities at the Process API layer. Effective decomposition strategies at this level identify clear structural "seams" within existing applications—natural boundaries where functionality separates with minimal disruption to interconnected components. These seams frequently correlate with varying change frequencies within the system, as certain clinical processes evolve rapidly, responding to emerging care models, while others maintain relative stability. The Process API layer implements several essential architectural patterns enabling effective monolith decomposition, including coordinated transaction management across distributed services, resilience mechanisms addressing dependent service failures, and patterns separating read and write operations, optimizing

Journal of Information Systems Engineering and Management
2025, 10(63s)
e-ISSN: 2468-4376
https://jisem-journal.com/

Research Article

performance characteristics. Healthcare organizations implementing Process APIs must address substantial database decomposition challenges, as clinical and administrative information frequently contains complex relationships spanning multiple bounded contexts. Successful implementations typically employ strategies including data replication with eventual consistency for non-critical information, while preserving transactional integrity within specific microservices where immediate consistency remains essential for patient safety. Process APIs fulfill crucial regulatory compliance functions during decomposition, implementing consistent rules enforcement regardless of whether functionality resides within legacy monoliths or new microservices [14]. This layer additionally addresses comprehensive audit trail maintenance within distributed architectures, implementing specialized logging and tracing mechanisms providing complete visibility into clinical and administrative processes spanning multiple services—capabilities essential for both regulatory compliance and clinical governance.

Experience APIs comprise the uppermost tier within the connectivity framework, delivering contextually appropriate information and functionality to diverse healthcare applications, including clinical workstations, patient engagement portals, mobile applications, and external partner systems. These APIs aggregate, transform, and filter information addressing specific consumption channel requirements, creating purpose-built interfaces delivering optimal user experiences without compromising security or compliance standards [15]. Experience API development benefits from insights gained through large-scale monolithic applications, where organizations discover that effective modularization provides numerous microservice advantages while avoiding certain distributed system complexities. Health care organizations discover that defining clear ownership boundaries in the Experience API layer instills accountability while allowing specialized teams to focus on particular user constituencies—e.g., clinician experiences, patient engagement, or partner integration. Such an ownership model often defines small, cross-functional teams that own entire lifecycles of particular Experience APIs, from the first design to ongoing evolution and support. Implementations frequently follow a capability-centered organization rather than traditional technical stratification, with each package containing all components necessary to deliver specific functionality regardless of underlying technology. This organization establishes natural boundaries for potential future decomposition as modernization initiatives progress. Experience APIs frequently leverage modular monolith approaches, maintaining deployment simplicity while achieving logical separation, using techniques including internal communication mechanisms between modules, feature activation controls managing functionality availability across user segments, and automated test boundaries maintaining clear interfaces between components [15]. This approach enables healthcare organizations to balance rapid development cycle benefits with governance requirements in healthcare environments, where regulatory changes may necessitate coordinated updates across multiple related capabilities.

API gateway implementation represents an essential architectural component within healthcare API strategies, providing centralized control for traffic management, security enforcement, and operational monitoring. Modern API gateways support incremental modernization patterns by functioning as facades intercepting and routing requests between legacy systems and new microservices based on configurable policies [13]. Infrastructure implementation of API gateways within healthcare environments requires specialized consideration, addressing both technical and operational requirements. Organizations undergoing modernization discover that implementing gateways as containerized applications within orchestration platforms provides essential capabilities for scaling, resilience, and consistent deployment across environments. Gateway architectures typically incorporate multiple specialized components, including traffic management mechanisms handling routing and load distribution, security enforcement layers implementing authentication and authorization, policy management frameworks enforcing rate limitations and compliance rules, and analytics engines capturing detailed metrics regarding API utilization and performance

Journal of Information Systems Engineering and Management
2025, 10(63s)
e-ISSN: 2468-4376
https://jisem-journal.com/

Research Article

characteristics. Healthcare implementations frequently deploy multi-tier gateway architectures with external-facing gateways managing external traffic, internal gateways coordinating service-to-service communication, and specialized security gateways implementing additional controls for particularly sensitive operations. This architecture provides defense-in-depth while enabling different security policies to be applied based on traffic sources and destinations. Operational models for gateway management frequently follow infrastructure-as-code principles, with all configuration managed within version control systems, enabling comprehensive audit trails documenting policy modifications and simplified recovery capabilities when issues emerge. Infrastructure automation plays an essential role in maintaining gateway security, with continuous vulnerability assessment, automated security credential rotation, and dynamic security policy updates, ensuring protection mechanisms remain current, addressing emerging threats [13]. This comprehensive infrastructure foundation enables API gateways to function as central coordination mechanisms for healthcare modernization initiatives, providing extensive visibility while enforcing consistent governance across both legacy and modern components within technology ecosystems.

The separation of responsibilities inherent within API-led connectivity delivers particular value within healthcare environments where protected health information requires specialized handling across system boundaries. By clearly delineating responsibilities across three API tiers, organizations establish consistent data protection practices aligned with security and compliance frameworks [14]. Monolithic healthcare system decomposition creates specific challenges in maintaining data security and privacy, as sensitive information previously contained within single application boundaries now traverses multiple microservices. Successful healthcare modernization initiatives address these challenges by implementing comprehensive data governance strategies across API tiers. These strategies begin with data classification, identifying sensitivity levels for different information types, from publicly shareable scheduling information to highly restricted genetic or behavioral health data. Based on these classifications, organizations implement progressive security controls across API tiers, with increasing protection for more sensitive information. These controls frequently include field-level encryption for particularly sensitive data elements, ensuring protection both during transmission and storage, regardless of which services process the information. Security implementations include specialized consent management capabilities tracking patient preferences for information sharing, and applying them consistently across distributed systems. Role-based access control mechanisms function as cross-cutting concerns across all API tiers, ensuring consistent user permission enforcement regardless of access pathways. Extensive audit logging provides permanent records of all access to protected health information, with dedicated monitoring systems interpreting these logs, detecting potentially inappropriate access or anomalous patterns potentially signaling security issues [14]. This multi-layered methodology of information governance allows healthcare organizations to preserve regulatory compliance and continuously modernize application portfolios while assuring security and privacy protections are vigorous even during transformational journeys.

Healthcare organizations must balance stringent compliance requirements with innovation capabilities, addressing rapidly evolving market demands and consumer expectations. The API-led connectivity framework supports this balance by creating controlled innovation zones where experimentation occurs without compromising core security and compliance foundations [15]. Organizations managing large-scale modernization initiatives discover that maintaining modular approaches within each API tier provides significant advantages, balancing stability and innovation. This approach begins by establishing clear boundaries between components, whether implemented as microservices or modules within structured monoliths. These boundaries create natural separation where different evolution rates accommodate varying needs, with critical clinical components maintaining appropriate stability while patient engagement and digital experience components evolve more rapidly. Governance models typically implement different change management processes based on component classification, with streamlined approval for low-risk changes while maintaining

comprehensive review for modifications affecting critical clinical or financial functions. Testing strategies play a crucial role in enabling controlled innovation, with comprehensive automated test suites creating safety mechanisms that detect unintended consequences before affecting production systems. These test suites typically include specialized validation addressing healthcare-specific concerns, including clinical decision support accuracy, regulatory compliance, and data consistency across distributed components. Development methodologies frequently employ progressive release mechanisms, introducing new capabilities to controlled user segments, enabling careful validation before widespread deployment. This approach proves particularly valuable for healthcare organizations balancing emerging technology introduction with maintaining essential reliability for core clinical functions [15]. By establishing clear boundaries with well-defined interfaces, whether through microservices or modular monoliths, healthcare organizations create technology landscapes evolving at different rates across different functional areas while maintaining overall system integrity and compliance—capabilities essential to remaining competitive within rapidly evolving healthcare marketplaces.

Conclusion

Transforming outdated healthcare platforms represents a multifaceted yet crucial undertaking for institutions seeking competitive viability while enhancing clinical results and operational performance. Applying architectural patterns and implementation strategies detailed throughout this article enables healthcare enterprises to successfully transition from unified architectures toward interface-prioritized, event-oriented ecosystems. The incremental replacement pattern delivers methodical frameworks for gradual substitution, reducing risk while preserving operational continuity. Domain-oriented design principles establish alignment between technical architecture and healthcare business domains, ensuring modernized systems accurately reflect clinical and administrative realities. The three-layered interface connectivity framework creates distinct separation between system access, process coordination, and experience delivery, establishing governance structures balancing stability, compliance, and innovation. Healthcare institutions embracing these patterns while adapting them toward specific regulatory and clinical requirements position themselves to incorporate emerging capabilities, including computational intelligence and remote monitoring, while preserving essential reliability for patient care. As healthcare pursues digital transformation, these architectural approaches create viable pathways for organizations to escape legacy constraints while preserving valuable institutional knowledge and maintaining uninterrupted service delivery throughout modernization initiatives.

References

[1] Stuti Dhruv, "Modernizing Legacy Systems in Healthcare," Aalpha, 2025. [Online]. Available: https://www.aalpha.net/blog/modernizing-legacy-systems-in-healthcare/
[2] Anagha Venugopal, "Healthcare Legacy System Modernization 2025: Types, Issues & Strategies," Dochours, 2025. [Online]. Available: https://dochours.com/healthcare-legacy-system-modernization/
[3] Tabby Ward and Dmitry Gulin, "Decomposing monoliths into microservices," AWS, 2023. [Online]. Available: https://docs.aws.amazon.com/prescriptive-guidance/latest/modernization-decomposing-monoliths/welcome.html
[4] James Lewis and Martin Fowler, "Microservices Guide," Martin Fowler, 2014. [Online]. Available: https://martinfowler.com/microservices/
[5] Sam Newman, "Monolith to Microservices," O'Reilly Media, Inc., 2019. [Online]. Available: https://www.oreilly.com/library/view/monolith-to-microservices/9781492047834/

[6] Martin Fowler, "Strangler Fig," 2024. [Online]. Available: https://martinfowler.com/bliki/StranglerFigApplication.html
[7] Greeksforgreeks, "Strangler Pattern in Microservices | System Design," 2023. [Online]. Available: https://www.geeksforgeeks.org/system-design/strangler-pattern-in-micro-services-system-design/
[8] Matt Tanner, "Seven application modernization case studies," vFunction, 2025. [Online]. Available: https://vfunction.com/blog/application-modernization-case-study/
[9] Greeksforgeeks, "Domain-Oriented Microservice Architecture," 2025. [Online]. Available: https://www.geeksforgeeks.org/system-design/domain-oriented-microservice-architecture/
[10] Tomas Fernandez, "Domain-Driven Design Principles for Microservices," 2022. [Online]. Available: https://semaphore.io/blog/domain-driven-design-microservices
[11] eInfochips PES, "MicroMigrate: Unveiling the Journey from Monolith to Microservices with Domain-Driven Design," 2025. [Online]. Available: https://www.einfochips.com/blog/micromigrate-unveiling-the-journey-from-monolith-to-microservices-with-domain-driven-design/
[12] Jon Edvald, "Seven Hard-Earned Lessons Learned Migrating a Monolith to Microservices," InfoQ, 2020. [Online]. Available: https://www.infoq.com/articles/lessons-learned-monolith-microservices/
[13] MaruthiTechlabs, "How We Made McQueen Autocorp's Systems Highly Scalable Using Kubernetes," 2025. [Online]. Available: https://marutitech.com/case-study/infrastructure-migration-to-kubernetes/
[14] Connectwise, "Decomposing the monolith: A guide to microservices transformation," 2024. [Online]. Available: https://www.connectwise.com/blog/decomposing-the-monolith-a-guide-to-microservices-transformation
[15] Shopify, "Under Deconstruction: The State of Shopify's Monolith," 2020. [Online]. Available: https://shopify.engineering/shopify-monolith

Reference.

I.A The Imperative for Modernization and Monolith Decomposition

1. Maruti Techlabs. (2025). *Case Study - Monolithic to Micros: Kubernetes Migration*. This case study details the migration of an automotive application with over 200 microservices from a single-node monolithic deployment to a highly available Kubernetes cluster. It highlights the use of the Strangler Fig pattern, containerization with Docker, and deployment across multiple availability zones to solve challenges like single points of failure and scalability limitations.
2. connectwise.com. (2024). *Decomposing the monolith: A guide to microservices transformation*. This guide provides a strategic overview of deconstruction, focusing on identifying "seams" or "bounded contexts" within the monolith. It discusses the critical challenges of database decomposition, managing shared data, and handling transactional boundaries in a distributed environment, recommending eventual consistency as a preferred model.
3. Labs Practices. (2023). *Deconstructing the Monolith: A Strategic Guide*. This guide outlines key principles for deciding when and how to decompose a monolith, such as considering multiple rates of change, independent life cycles, and failure isolation. It introduces "Event Storming" as a collaborative technique to discover bounded contexts and identify candidate microservices by mapping business events.