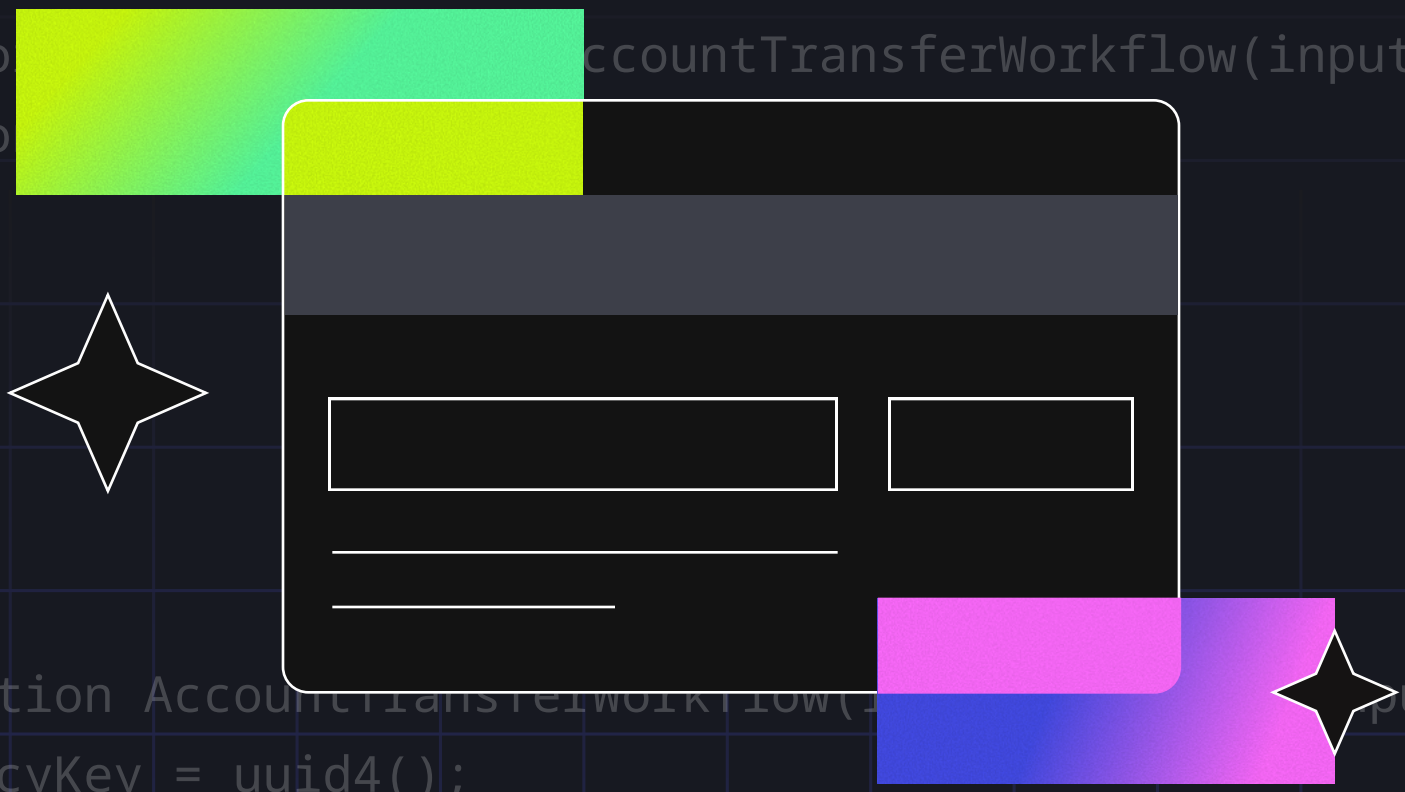
Temporal



A Guide to Modernizing Systems in Financial Services

Table of Contents

The Challenge of Modernizing in Financial Services 4

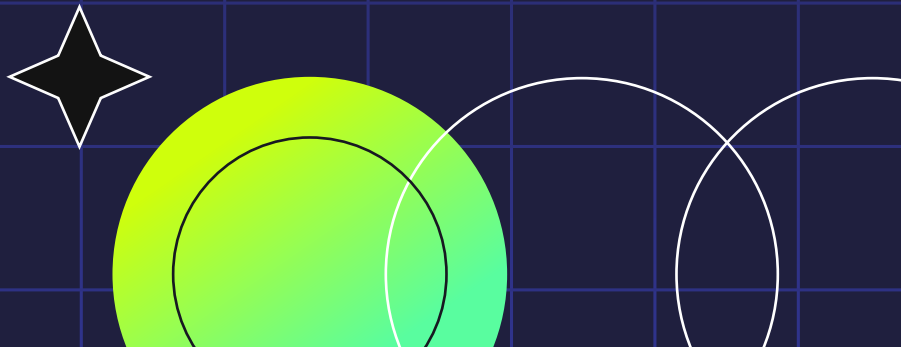
- Reliance on Monolithic Architecture 4
- Managing Workflow Complexity 5
- Replacing Legacy Orchestration Tools 5

Simplifying Modernization with Temporal 6

Payment Workflow Architecture 7

Case Study: ANZ Bank 8

Preparing for What's Next 9



Only...

35%

of banking customers,

34%

of insurance customers, &

39%

of wealth management customers

are fully satisfied with their financial institution's digital experience.

Financial institutions continue to invest heavily in digital transformation, modernization initiatives, and AI-powered services. Yet customer expectations continue to rise, and satisfaction with digital experiences remains relatively low.

According to the 2025 Salesforce Connected Financial Services Report, only 35% of banking customers, 34% of insurance customers, and 39% of wealth management customers report being fully satisfied with their institution's digital experience.

For engineering and architecture leaders, delivering a better customer journey is no longer simply a matter of launching new applications or digital channels. The systems behind those experiences must be reliable, scalable, and capable of supporting increasingly complex workflows.

Customers may never see the systems behind the experience, but they still expect every interaction to be seamless. Whether they are applying for a loan, opening an account, or moving money, they need those interactions to work consistently and without interruption.

At the same time, financial institutions are expanding digital capabilities, introducing AI-powered services, and modernizing core systems that have evolved over decades. These initiatives create new opportunities for growth and innovation, but they also increase the complexity of the systems and workflows that power the business.

Meeting these expectations requires more than investing in new technology. As organizations modernize and replace legacy infrastructure, the workflows behind every customer touchpoint must continue to operate reliably at scale.

The Challenge of Modernizing in Financial Services

Financial institutions are transforming how they deliver customer experiences through digital services, connected data, and AI-powered capabilities. But every new application, integration, and workflow increases the complexity of the systems supporting those experiences.

Reducing operational risk starts with understanding where that complexity comes from. Legacy architectures, distributed workflows, and outdated orchestration tools can all slow modernization efforts, making systems harder to scale, maintain, and recover when failures occur. Addressing these challenges is the first step toward building resilient, reliable applications that support innovation.

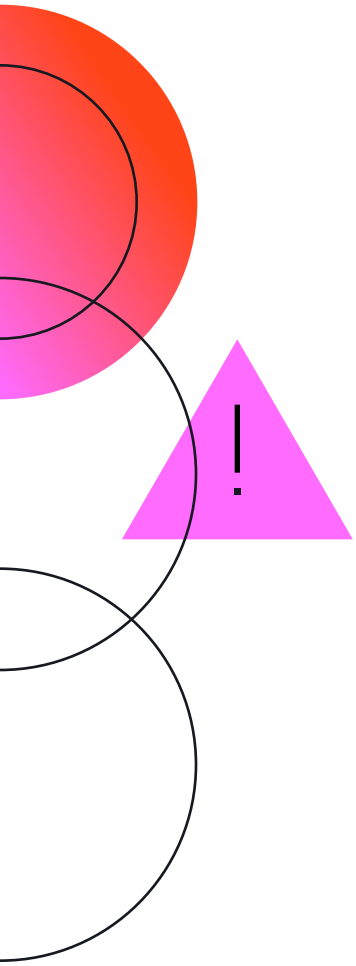
Reliance on Monolithic Architecture

Many financial institutions still rely on monolithic applications to power their most critical business processes. Modernizing these systems without disrupting customer experiences is one of the biggest challenges engineering teams face.

By nature, monolithic architectures are difficult to maintain and accumulate technical debt, making them susceptible to outages and security vulnerabilities. Their tightly coupled structures reduce agility, create deployment bottlenecks, and slow teams' ability to respond to changing business and customer needs.

The limitations of monolithic systems extend beyond day-to-day operations. As organizations modernize and introduce new digital services, these systems often require extensive reworking just to implement minor changes. This slows innovation, increases development effort, and makes it harder to continuously improve customer experiences.

Scaling also becomes a significant hurdle. As transaction volumes grow and new services are introduced, monolithic architectures struggle to keep pace. Adopting more modular architectures enables organizations to modernize incrementally while improving scalability and long-term resilience.



Managing Workflow Complexity

Many financial institutions have evolved from monolithic architectures to event-driven architectures (EDA) as part of their modernization efforts. While this approach improves flexibility, choreography-based EDA can introduce significant complexity that makes systems more difficult to manage, troubleshoot, and scale.

Unlike orchestration, where a central coordinator manages workflow execution, choreography relies on individual services to independently react to events and trigger subsequent actions. As workflows span more services, dependencies become harder to understand, increasing the likelihood of failures and making it more difficult to maintain consistent business processes.

Managing event-driven systems at scale introduces additional complexity. Business logic becomes scattered across multiple services, making it difficult for developers to understand how workflows execute across the system. Troubleshooting often requires tracing a single transaction across numerous services and logs, slowing issue resolution and reducing operational visibility.

Organizations looking to simplify distributed systems often move toward centralized workflow orchestration. This approach improves visibility, simplifies coordination, and reduces operational complexity while making systems easier to scale and recover.

Replacing Legacy Orchestration Tools

Many organizations adopted Business Process Modeling Notation (BPMN) platforms and legacy orchestration tools to manage increasingly complex workflows. While these solutions helped standardize business processes, they often introduced new development and operational challenges as systems evolved.

One of the main drawbacks of these systems is that they are not code-first. They rely on rigid, non-programmatic models, limiting developers' ability to build, test, and innovate. As workflows grow more complex, business logic becomes hidden behind proprietary interfaces, making systems more difficult to understand, maintain, and troubleshoot.

These limitations slow development, increase operational overhead, and make modernization more difficult. Modern engineering teams increasingly need workflow orchestration that integrates naturally into existing development practices, enabling faster innovation without sacrificing reliability.

Organizations using Temporal Cloud achieved:

50%

faster time to market

15%

improvement in developer productivity

201%

return on investment

Simplifying Modernization with Temporal

Modernizing systems doesn't have to mean replacing everything at once. Temporal provides a Durable Execution platform that helps engineering teams simplify complex workflows while improving reliability, scalability, and developer productivity.

Rather than requiring organizations to rebuild orchestration, state management, retries, and failure handling from scratch, Temporal manages those complexities automatically. Developers can focus on business logic using familiar programming languages while Temporal ensures workflows execute reliably even when applications, infrastructure, or external services fail.

This approach not only reduces operational complexity but also delivers measurable business outcomes. According to the Forrester Total Economic Impact Report, organizations using Temporal Cloud achieved:

- **50%** faster time to market
- **15%** improvement in developer productivity
- **201%** return on investment over three years

By eliminating the need for brittle custom orchestration and simplifying distributed application development, Temporal enables organizations to modernize incrementally while continuing to support the mission-critical systems customers rely on every day.

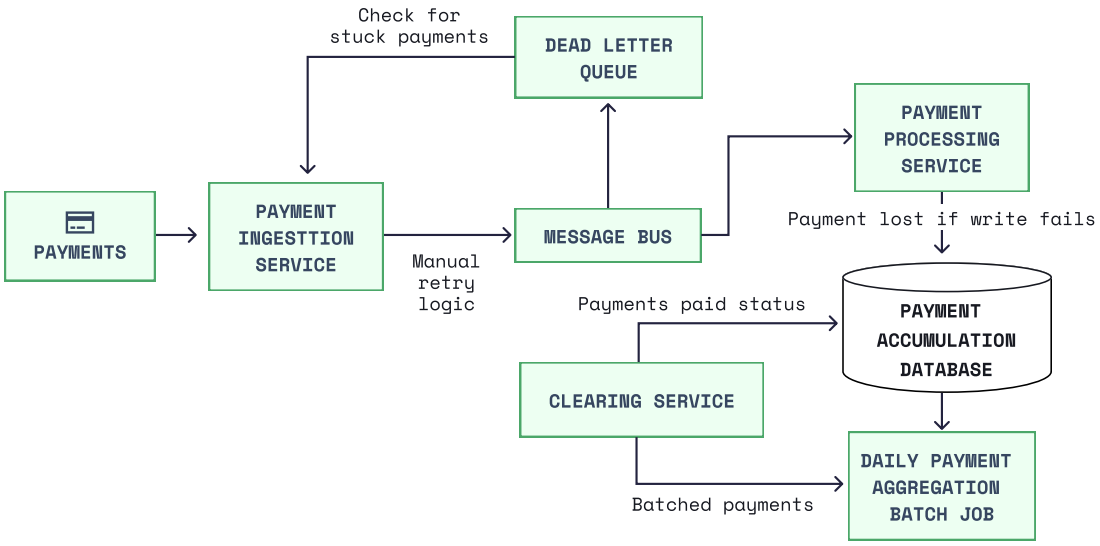
The following example illustrates how Temporal simplifies a complex payment workflow, reducing architectural complexity while improving reliability and developer productivity.

Payment Workflow Architecture

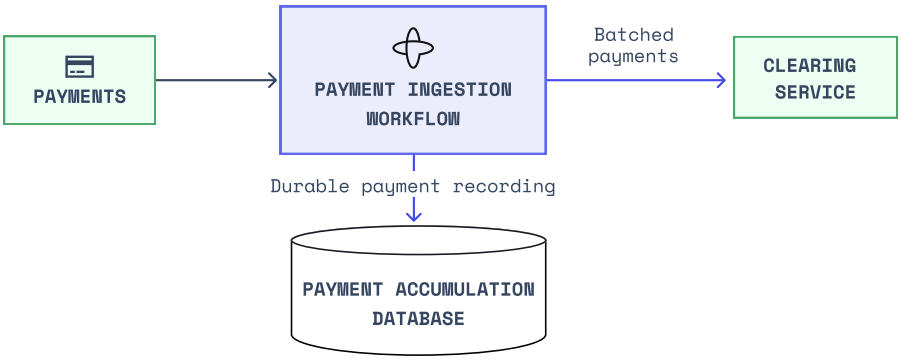
Many payment workflows rely on a combination of event queues, custom retry logic, distributed state management, and multiple services working together to process a single transaction. While this architecture can scale, it also introduces operational complexity that makes systems more difficult to build, maintain, and troubleshoot.

As workflows become more distributed, engineering teams must manage additional infrastructure, recover from failures manually, and maintain custom code to ensure payments are processed reliably. Each new component increases the number of potential failure points, slowing development and increasing operational overhead. The following example illustrates how complexity accumulates in a traditional event-driven payment architecture.

Before Temporal:



After Temporal:



Case Study: ANZ Bank

ANZ, one of the largest banks in Australia and New Zealand, demonstrates how financial services organizations use Temporal to modernize mission-critical systems.

One of ANZ's most ambitious modernization initiatives was rebuilding its home loan origination system, which required a complete re-architecture of legacy systems. The ANZ team faced this challenge for over 18 months before adopting Temporal. Within six weeks, ANZ had its home loan origination system in production, accelerating project delivery by 8× compared to its previous pace.

> "Over the course of a single weekend, we achieved with Temporal what had eluded us for an entire year."
>
> — ANZ Team

According to the team at ANZ, "Over the course of a single weekend, we achieved with Temporal what had eluded us for an entire year."

Temporal addressed several critical challenges for ANZ's modernization efforts:

Error-handling abstraction: Developers could focus on business logic without needing to manage error-handling manually.

Automated compensations and state management: Temporal ensured all processes were completed in the correct order or rolled back automatically.

Smooth integration with third-party APIs: Temporal's support for human-in-the-loop interactions and configurable retry policies allowed seamless integration with systems like Salesforce, preventing process timeouts.

By allowing ANZ to streamline complex workflows, reduce manual intervention, and improve project delivery speed, Temporal facilitated a scalable, domain-driven design for the team. This freed up time for ANZ to focus on delivering great new services while reducing complexity.

With Temporal, ANZ achieved project delivery

8x

faster than before.

Preparing for What's Next

Financial services organizations are entering a new era of transformation. AI-powered experiences, real-time payments, and evolving customer expectations are accelerating the pace of innovation. At the same time, engineering teams are expected to modernize critical systems without compromising the reliability their customers depend on.

The challenge isn't simply adopting new technologies, it's building an architecture that can support continuous change.

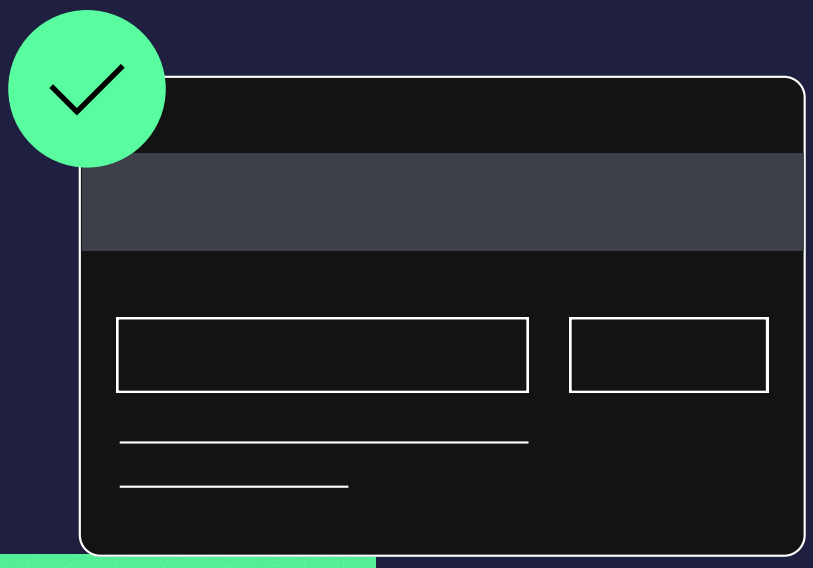
Organizations that succeed won't be the ones that replace every legacy system first. They'll be the ones that simplify complexity and give engineering teams the foundation to build with confidence.

Temporal provides that foundation. By simplifying workflow orchestration and abstracting the complexity of distributed systems, Temporal enables teams to modernize mission-critical applications while improving developer productivity and long-term scalability.

Whether you're building the next generation of payment systems, modernizing loan origination, or deploying AI-powered services, Durable Execution helps ensure your workflows remain reliable as your business evolves.

The future of financial services will be defined by the organizations that can innovate without sacrificing reliability. Preparing for what's next starts with building systems that are designed to evolve.

Explore how Temporal can help your FinServ team build resilient, future-ready applications.



Temporal

JOIN OUR COMMUNITY SLACK



EXPLORE PROJECT-BASED TUTORIALS



AND DOCS

