International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 10 October 2026
Page No: 5937 - 5957
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0763

Intelligent Core Banking Transformation: Building Scalable, Resilient, and AI-Ready Digital Banking Ecosystems

Chaganty Vijay Kumar[1]
[1]*Independent Researcher Andhra University Visakhapatnam, India*
Email: *vijay.chaganty@gmail.com*[1]

Abstract
Core banking systems serve as the official record for almost all regulated financial institutions, and in the case of most large banks the system used for this purpose is still based on a monolithic, outdated platform—often COBOL running on a mainframe—rather than being replaced, the system being wrapped in a series of point-to-point integrations. Modernization is not a choice: real-time payment networks based on ISO 20022, open-banking APIs, and customer expectations shaped by digital-native fintech companies have all moved beyond the capabilities of a batch-processing core, and the generative and agentic AI capabilities currently transforming all the other areas of the business cannot safely be incorporated into a core that itself cannot provide clean, governed, real-time data and actions from the outset. Nevertheless, core banking transformation has one of the poorest records for delivery of any type of enterprise technology project: industry analyses show that only about 30% of such transformations successfully complete the full migration of both the ledger and the product range, that cost overruns of up to 100% and delays in the timeline of 50 to 100% are common, and that even those programmes that are considered to have been largely successful still leave 10 to 20% of the old functionality running more than five years after going live. This paper contends that this pattern of failure is due to architecture, not just to management: most transformation initiatives treat modernization (involving the decomposition of the monolith), integration (involving the exposure of APIs and events), resilience (in meeting regulatory requirements for operational resilience), and AI-readiness (making the core safely usable by AI and agentic systems) as four separate activities managed by four different teams, rather than as part of a single system that must be designed together right from the beginning. We propose CORE, a reference architecture centred on four integrated engines—Cloud-native Modernization, Open API and event Orchestration, Resilience and continuity, and Embedded intelligence—that share one data and event fabric and one governance and observability layer, so that a bank's migration wave plan, its API and event contracts, its resilience position, and its AI autonomy policy are all designed as a single decision rather than being settled afterwards. We outline CORE's five-layer architecture, provide an example of how capabilities are broken down and describe a matrix for tiering AI autonomy that maps the existing core capabilities to the target microservices and specifies how much autonomy is granted to the embedded AI for each, include a case study that shows how a composable deposits-and-payments core can be migrated using a strangler-fig approach under a surge in real-time payment demand, present an evaluation plan based on comparable published benchmarks, and address the issues relating to deployment, regulation, and ethics when running such a programme within a globally regulated bank. The contribution is a practical reference architecture derived from experience in core banking transformation, offered as a blueprint for architects and technology leaders who are planning agentic, AI-ready core banking programmes and as a foundation for future empirical validation using data from actual production transformations.

Keywords: *core banking modernization; composable banking; cloud-native architecture; API-first banking; event-driven architecture; operational resilience; site reliability engineering; agentic AI; explainable AI; data mesh; ISO 20022; digital banking transformation.*

1. Introduction
The core banking system is the single system that underlies all other banking functions: it is the one which records a deposit, the one that calculates interest, the one that decides if an overdraft is allowed, and it is the definitive source of truth that must be agreed upon simultaneously by a mobile app, a branch teller, and a real-time payment system. For most large and well-established banks, this system of record is a monolithic platform that was originally set

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 10 October 2026
Page No: 5937 - 5957
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0763

up during the 1970s and 1990s, typically operating COBOL on a mainframe, and has been extended over the years by successive generations of engineers through direct changes to the code and by the addition of a growing number of point-to-point batch interfaces rather than by adopting a consistent integration architecture [1], [2]. While this by itself is not a crisis—mainframe COBOL systems are still exceptionally reliable and handle a large portion of the world's daily transaction volume without any problems—the real problem is that the kind of environment in which these systems were developed no longer exists. The real-time payment systems—such as the U.S. FedNow Service and RTP, the UK's Faster Payments, India's UPI, and the European Union's instant SEPA credit transfer scheme—are based on the ISO 20022 structured-data messaging standard and require settlement to take place within seconds, not after the overnight batch process that the old core systems were designed for [3], [4]. Regulations relating to open banking and account aggregation in the UK, the EU (PSD2, with PSD3 negotiations broadening it further), and an increasing number of other regions now require banks to make their account and payment-initiation data available via standard APIs to licensed third parties, not just to their own channels. Furthermore, customer expectations—shaped by digital-native challenger banks and fintechs that have always been built with an API-first approach—now regard real-time balance updates, instant account opening, and personal, AI-driven service as something that should be the standard rather than something that sets a business apart. Generative and agentic AI actually increases this pressure rather than reducing it. An intelligent large language model agent which prepares a summary of a fraud investigation, suggests a personalised product offer, or automatically approves a routine service request is just as useful — and just as safe — as the data it can access and the actions it is able to carry out. An agent positioned in front of an outdated core system that makes data available only through nightly batch extracts and carries out actions via a small number of stiff, old transaction codes from decades ago cannot reason about current balances, cannot carry out a limited action within the ledger without a human having to enter it through the same green-screen interface that a teller uses, and cannot be given an auditable and reversible action space as a modern, API-based system can by design. A study of the agentic-AI situation in banking in 2026 states that there is an ever-widening divide between the banks that are testing agentic features in narrow and well-monitored areas and the much larger number of banks whose core infrastructure still cannot support such monitoring at all [5], [6], [7]. To put it plainly, 'AI-ready' is not a separate project added on after core modernisation; it is a feature that the modernised core must be designed to have. Even though there is this pressure, core banking transformation has one of the worst records when it comes to delivering any of the major categories of enterprise technology programs. Analysis that has been published on core banking system transformations shows that only about 30% of such transformations successfully carry out a full migration of the ledger and products to the target platform; that cost overruns reaching as much as 100% and delays in the timetable of 50 to 100% compared to the original plan are common; that stakeholders usually underestimate the amount of work involved in the migration by as much as 75%; and that even those programs which are regarded as broadly successful often still have 10 to 20% of the old system's functionality running on the original platform more than five years after the new core system first goes live, since the most difficult and high-risk products and customer groups were postponed rather than migrated [1]. The same analysis also reveals that successful transformations tend to leave about half of the performance and efficiency gains that a modern core system is capable of achieving unrealized, since the new platform is deployed and operated using the same methods — waterfall planning, siloed operations teams, and minimal automation — as those that were used to manage the old system it replaced [1]. The problems are not primarily technological, per se; in most instances, the failure or halt in the process of transformation can be blamed on the functioning of the platforms to which it is addressed (the target platforms). Instead, they represent architectural and organizational failures—the decomposition and integration and resilience and (now more than ever) AI-readiness was planned and managed as a simple technology replacement project. Structural fragmentation is reflected in the organisation of transformation programmes. The modernization workstream is usually led by an application-

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 10 October 2026
Page No: 5937 - 5957
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0763

architecture team and is focused on figuring out how to decompose the monolith into services, usually through the steps of strangler-fig, where over time traffic is diverted from the monolith to the new services, until the original system can be retired [8], [9]. The integration workstream, usually a different API / middleware team, creates the API and event contracts to enable the new services to be accessed by various channels and partners using, for example, the Banking Industry Architecture Network (BIAN) service landscape or the "coreless banking" pattern where the best available services are used around a minimal ledger core [10] and, in order to cope with the modernisation of the payment infrastructure, is now required to provide interfaces in ISO 20022-native and real-time [3] [4]. The resilience and operations workstream, usually within a separate risk or infrastructure function, is responsible for ensuring that banks meet the operational-resilience requirements formally set out by bank supervisors: the Basel Committee's 2021 Principles for Operational Resilience (POR) require banks to identify critical operations, define tolerances for disruption and test their ability to remain within those tolerances using severe but plausible scenarios [11]; in the EU, the Digital Operational Resilience Act (DORA) imposes directly enforceable obligations on banks and critical technology providers [12]. Furthermore, a fourth workstream, for AI or data, is more and more on the task of getting the bank's data ready to be used by AI, and often, many years after the architectural decisions of the previous three workstreams have been made and are therefore difficult to change. These are all four workstreams that are capable of what they are doing. None of them, in isolation, could be responsible for recognizing that the scope of the impact of the modernization workstream has been selected for engineering convenience, that the API contract designed without consideration for the granularity of events in the system would limit the AI workstream on what it can safely observe and act upon years later, or that a migration wave ordered solely based on technical dependency, and not business risk, is why the 10–20% of legacy functionality has been left behind and why published analysis identifies it as one of the more common failure signatures [1]. The manuscript presents three contributions. The first of these is the introduction of CORE, a reference architecture which arranges the intelligent transformation of core banking into four integrated engines — Cloud-native Modernization, Open API and event Orchestration, Resilience and continuity, and Embedded intelligence — all of them sharing a single data and event fabric as well as a single governance and observability layer, so that decomposition, integration, resilience, and AI-readiness are designed as part of a unified system rather than being addressed separately in four different workstreams afterwards. The second contribution is the specification of an illustrative capability-decomposition and AI-autonomy tiering matrix which links legacy core capabilities to the target microservices, to strangler-fig migration patterns, and — most important of all — sets the autonomy tier of the AI that is granted over the resulting service, this autonomy being calibrated according to the reversibility and blast radius of the action and not being applied uniformly throughout the entire environment. The third contribution consists of an illustrative case study concerning a composable deposits-and-payments core that undergoes a strangler-fig wave plan while at the same time absorbing a surge in real-time payment demand, demonstrating how the four engines work together to decompose, expose, protect, and intelligently operate the resulting platform. Finally, the manuscript includes considerations regarding deployment, regulation, and change management when introducing such a programme within a globally regulated bank, along with an ethical and governance discussion that is suitable for a system capable of taking autonomous action with respect to the bank's ledger, and a frank assessment of the framework's limitations.

2. Related Work

2.1. Strategies for the modernization of core banking and the patterns involved in migrating legacy systems

The amount of writing both by practitioners and in the field of industry analysis on core banking transformation is vast and mostly arrives at the same conclusion: although it is an organizational change, transformation is seen as a move to new technology. Analysis that has been published on the results of such transformations records the specific pattern of failure which this paper takes as the basis for its motivation — namely, a full-migration success rate of about 30%, with cost and time overruns frequently

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 10 October 2026
Page No: 5937 - 5957
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0763

amounting to 50–100%, and a continuing number of legacy functions that have not been migrated in programmes that are otherwise considered successful — and assigns this to a repeated series of mistakes relating to people, processes, and technology: the delivery being led by operations teams rather than by special transformation teams, the use of long waterfall planning cycles, true costs being obscured, and vendor selection being based on cost rather than on a good fit with the banking domain [1]. Another body of practitioner literature explains why the old platforms continue to exist even in the face of this pressure: mainframe cores based on COBOL remain reliable, fast, and are very deeply embedded in decades of accumulated business logic that is both expensive and risky to re-create, and at the same time the number of engineers who can maintain that logic is decreasing because COBOL-trained staff are retiring faster than they are being replaced [2]. This body of literature sets out the baseline for the delivery and outcomes that CORE is designed to address and provides the "reason why it is not simply advisable to replace it" that justifies adopting an incremental, strangler-fig style approach instead of a big-bang rewrite.

2.2. The BIAN Reference Model and Composable Banking Platforms

The enterprise-architecture standards community as well as the core banking vendor market have both arrived at a "coreless" or composable approach as the desired end-state architecture: instead of having a single monolithic core that covers all banking functions, a slim ledger core is surrounded by individual, best-of-breed services which can be independently deployed for the purposes of origination, payments, cards, and channel experience, the various services being integrated via standard service contracts. BIAN, the Banking Industry Architecture Network, has established this approach through its semantic API specifications and its "Coreless Banking" reference deliverable, providing architects with a vendor-neutral service environment and API terminology that they can use to break down their systems rather than having to create service boundaries from scratch in each instance [10]. The core banking vendors have also adopted this pattern directly: Temenos provides composable banking services on its cloud platform so that individual product or channel features can be taken on gradually rather than requiring a complete platform swap [13], and Mambu positions itself explicitly as a composable, API-first core banking platform that is meant to be put together from independently replaceable components rather than being deployed as a monolith [14]. The body of literature and the range of product offerings available now define what a modernized target architecture should be; however, they say very little about the migration process — that is to say, how a bank can move from its current monolith to that target state without causing an unacceptable disruption to its live, regulated ledger — a gap which the next section of the strangler-fig migration literature seeks to fill.

2.3. Incremental Migration: The Strangler-Fig Pattern

The strangler-fig approach — that is, gradually diverting traffic for a particular capability from the old system to a new service before finally retiring the old one, rather than making a single switch-over — has become the most widely used method for reducing the risks associated with this type of migration, and the examples provided in the reference implementations not only describe the mechanics of the routing (such as the use of a facade or proxy layer that enables individual capabilities to be redirected) but also outline the governance procedures needed to actually disable the old system route once traffic has been transferred, rather than letting it continue to run indefinitely as dead but still active code [8]. A case study that puts this approach into practice within an actual production environment shows the method's real-world value: the various capabilities were migrated one at a time behind a routing facade, with the original implementation retained as a backup until the new service had been proven reliable under actual production load, thus substantially lowering the risk compared with a single cutover [9]. Although the migration techniques described in this literature form the basis of the CORE Cloud-native Modernization Engine, they have been generalised to apply to any kind of legacy system; the solution does not take into account the specific needs of core banking — for example, carrying out dual-run reconciliation against a regulated ledger and ensuring real-time consistency across the payment network during the migration process — which must be added to the generic pattern as part of a bank's transformation programme.

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 10 October 2026
Page No: 5937 - 5957
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0763

2.4. The modernization of the real-time payment rail, the adoption of an API-first integration approach, and the implementation of an event-driven architecture in the banking sector

An increasing amount of literature is now focusing on the use of event-driven microservices and cloud-native infrastructure in relation to banking systems, claiming that it is the ability to propagate events asynchronously and with loose coupling that enables a modernized core to satisfy both the high transaction volumes and the fast deployment speeds that traditional batch architectures are unable to achieve, while at the same time providing the fault isolation needed by a regulated and mission-critical ledger [15]. Since a composable core that exposes APIs increases the attack surface compared to a mainframe that can only be accessed via strictly controlled internal channels, there has emerged a parallel and equally pressing body of work concerned with the security of API gateways in the context of digital banking, suggesting the adoption of zero-trust architectures in which every API call—including traffic between internal services—is authenticated and authorized rather than being automatically trusted based on its network location [16], [17]. The real-time payment infrastructure introduces an additional integration requirement that is time-bound upon this event-driven foundation: both the U.S. Federal Reserve's FedNow Service and the RTP network use the ISO 20022 structured-data messaging standard, which includes much more detailed information about remittances and counterparties than the older payment message formats it replaces and which therefore requires a core banking platform to validate, route, and post a payment within a settlement window measured in seconds rather than during the batch cycle that older cores were built to handle [3]. The Federal Reserve System's analysis of this transition views the adoption of ISO 20022 not as a simple upgrade of the messaging format but as the standard that forms the basis of the wider effort to modernize the payments system, since a core that is unable to natively read and produce ISO 20022 messages ends up moving the translation and reconciliation logic into a fragile boundary layer instead of keeping it within the core itself [4]. This body of literature provides the integration-layer and real-time payment-rail mechanisms that CORE's Open API and Event Orchestration Engine is based on; it usually concentrates on the integration layer and the payment-rail standard separately, and does not link the design decisions regarding APIs, events, and the messaging standard back to the resilience requirements or the AI-observability needs that those same contracts must ultimately meet, nor does it deal with how a bank should sequence its overall core migration so that native ISO 20022 connectivity is available by the time that a particular payment rail requires it.

2.5. Resilience Engineering, Site Reliability Engineering, and Chaos Engineering in the financial services sector

The requirement for operational resilience in the financial sector has changed from being a matter of preference within engineering teams to becoming an official expectation enforced by regulators. The Basel Committee on Banking Supervision's Principles for Operational Resilience require banks to determine which of their operations are critical, establish clear limits on the level of disruption those operations can tolerate, and carry out tests—rather than just keeping records—of their ability to stay within those limits when facing severe but possible scenarios such as cyber attacks, technological failures, and disruptions caused by third parties or outsourcing [11]. The Digital Operational Resilience Act (DORA) is a principle-based requirement, transposed into directly enforceable EU legislation, imposing binding obligations on banks for information and communication technology (ICT) risk management, reporting of incidents and resilience testing as well as for key third-party technology providers, including the cloud and platform providers essential to a modernization program like CORE [12]. On the engineering side, chaos engineering is an approach that consists of intentionally injecting failures into a system to test the validity of its resilience assumptions before the failures happen unexpectedly during actual events; it has evolved from an experimental discipline to a well documented and tool-supported one, and an empirical study of the adoption of chaos-engineering tools in nearly a thousand open-source repositories reveals that there has been steady growth in both the maturity and the level of adoption of these tools, with the majority of the use cases being around fault injection at the network level and at the instance termination level, while fault injection at the application level is still underrepresented [18]. One such application of this discipline to the financial

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 10 October 2026
Page No: 5937 - 5957
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0763

services industry refers to the importance of the distinction between a controlled chaos experiment and an uncontrolled incident in a production environment, especially in a field where an outage has direct regulatory and reputational consequences [19]. A separate body of literature on deployment architectures looks at the specific pattern that banks are now using to achieve strict availability targets: active-active deployment across multiple cloud regions, so that when a failure occurs in one region it is handled by the continued operation of a healthy region rather than triggering a slower failover and recovery process [20]. This body of literature provides the resilience mechanisms that form the basis of CORE's Resilience and Continuity Engine; it is generally developed independently of the modernization and integration literature mentioned above, leaving unanswered the question of how a bank should decide, on a capability by capability basis, which services are worth the investment in active-active, multi-region setups and which are not—a question that this paper argues should be guided by the same blast-radius reasoning that is applied to both migration sequencing and AI autonomy (Section 4.3, Table 2).

2.6. Agentic and Generative AI in the field of banking operations

The way in which analysts and practitioners view the introduction of agentic AI into the banking sector highlights both the magnitude of the opportunity and the seriousness of the governance gap that it creates. Industry analysis states that 2026 will see agentic AI in banking and investment services go from being a pilot project to becoming a real issue for chief information officers concerning the overall operating model, while at the same time warning that the gap in technology readiness between banks which have already modernised their core data and action interfaces and those which have not is now the main factor in determining which banks will even be able to safely adopt agentic capabilities [5], [6]. A similar industry survey looking at AI trust in the 2026 "agentic era" shows that organisations' confidence in autonomous AI actions has not kept up with the ambition of deploying them, and that it is trust—rather than the raw capability of the model—that is becoming the key limitation on how much autonomy companies are willing to allow [7]. The AI governance documents released by individual banks reflect the pragmatic response to this gap, describing explainability, human oversight and model risk management as necessary prerequisites for scaling AI in banking instead of compliance requirements bolted on at the end [21]. The agent architecture literature that this coverage is based on (the ReAct approach of alternating explicit reasoning and tool-calling actions [22] and the evidence that breaking down a task into a set of coordinating specialised agents performs substantially better than a single monolithic agent when carrying out operationally realistic, multi-step tasks [23]) is general across domains; it provides the orchestration patterns that CORE's Embedded Intelligence Engine is built on, but it does not cover the core requirement of the banking domain that every action performed by an agent must boil down to a governed, reversible ledger operation.

2.7. Explainable AI for Credit Risk and Fraud Detection

AI models are employed by banks and financial institutions to determine eligibility for loans or credit cards, and to detect potential fraud. Such models can be quite complicated and it is difficult to understand how they make their decisions. Explainable AI (XAI) helps by providing the key elements that contribute to a specific prediction. This allows banks to review the model to determine whether the model is fair and rule compliant. It also provides feedback to the customer on the reason for loan decline/flag. Examples of common XAI techniques are SHAP and LIME. These tools are able to dissect a model's prediction and indicate what is most important. This not only increases the trust and insight of both companies and customers, but also helps streamline the decision-making process. In banking application, where an unexplained model score is least acceptable, there is a large body of literature using explainable AI (XAI) techniques in two areas: credit risk and fraud detection. Cloud services and deployment models as defined in NIST Special Publication 800-145 is the definition that most banking risk and compliance departments look at when deciding the regulatory responsibilities that accrue to a given core banking deployment [28] and the joint statement released by the U.S. federal banking agencies on security in a cloud computing environment makes clear the supervisory requirements that a bank must meet when running core systems on a cloud provider's infrastructure but not within its own data centre [29].

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 10 October 2026
Page No: 5937 - 5957
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0763

Recent research, which combines gradient-boosted ensembles with SHAP and LIME attribution methods when assessing credit risk, shows that it is possible to achieve explainability without sacrificing predictive performance, since tree-ensemble models are still able to provide strong accuracy together with personalized, auditable explanations for each lending decision [24]. At the same time, other work has used explainable AI specifically in the context of large-scale fraud detection and has looked at stacking-ensemble architectures that integrate several different model families with post-hoc attribution in order to balance detection accuracy against the need for interpretability in a fraud investigation case file [25], [26]. This body of literature forms the basis for the explanation methods that CORE's Embedded Intelligence Engine uses when dealing with its own risk-scoring outputs; these methods are generally developed using a fixed and already available data feed and do not tackle the earlier architectural question that is at the heart of this manuscript—whether the underlying core banking platform can make the real-time, well-governed data that these models require available at inference time.

2.8. The data architecture aimed at AI-readiness: data mesh and data fabric

The extent to which a core banking platform is ready for AI depends just as much on how its data is structured for use as on how its services are broken down for execution. When the data mesh approach—is characterized by domain-oriented design and federated data ownership with each domain acting as a publisher of its data as a governed product—is compared with the data fabric approach—which involves a single, metadata-driven layer for access and governance that covers diverse data sources—it is found that most large enterprises, including banks, end up needing aspects of both: domain ownership in order to keep data quality accountable to the team that understands it, and a fabric-type access layer so that an AI system (or an analyst) does not have to know which of the numerous systems owned by different domains a particular piece of data is located in before it can be used [27]. This body of literature is generally developed separately from the core-modernization literature in Sections 2.1–2.3; it assumes that the data layer can be redesigned on its own timetable, rather than viewing it as a feature that is determined years in advance by the decomposition boundaries and the API/event contracts of the core.

2.9. The regulatory and standards environment relating to core banking hosted in the cloud

The fact that a bank transfers its system of record to cloud infrastructure is not just an internal architectural choice but also a matter that is formally regulated and governed in accordance with established standards. As shown in Table 1 A comparative summary of the related work in relation to the capabilities that CORE integrates. The definition of cloud services and deployment models set out in NIST Special Publication 800-145 is the one that most banking risk and compliance departments still refer to when determining the regulatory responsibilities associated with a particular core banking deployment [28], and the joint statement issued by the U.S. federal banking agencies on security in a cloud computing environment establishes clear supervisory requirements regarding how a bank must manage identity and access control, configuration, and logging when its core systems are operated on a cloud provider's infrastructure rather than in its own data centre [29]. This regulatory guidance, combined with the operational-resilience duties mentioned in Section 2.5, is what leads CORE to place a strong emphasis on maintaining an auditable evidence trail that includes explanations for decisions relating to migration, integration, resilience, and the use of AI, rather than treating governance as a separate compliance activity that is added on afterwards once the architecture has already been put in place.

2.10. Research Gap

Table 1 presents the literature in relation to five capabilities that are relevant to the process of architecting an intelligent core banking transformation specifically. There is no single approach that combines (a) an incremental, strangler-fig-based decomposition approach based on a standard service reference model such as BIAN, (b) API- and event-native integration designed for real-time operations and aligned with ISO 20022 standards as well as featuring zero-trust exposure, (c) resilience engineering tuned to formal operational-resilience requirements rather than to general availability targets, (d) an AI autonomy model which governs embedded intelligence on a per-action basis rather than on a per-system basis, and (e) a common, audit-

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 10 October 2026
Page No: 5937 - 5957
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0763

defensible explanation layer covering all of the above aspects.

Table 1 A comparative summary of the related work in relation to the capabilities that CORE integrates

Related Work Area	Focus	Technique	Explainability	Gap CORE Addresses
Core transformation outcome analysis [1], [2]	Delivery track record and root-cause diagnosis	Program-level retrospective analysis	Low (organizational, not architectural)	Diagnoses the failure pattern but does not prescribe an integrated architecture
Composable banking / BIAN [10], [13], [14]	Target-state coreless architecture	Reference service model, vendor platforms	Platform-native	Defines the destination; largely silent on migration mechanics
Strangler-fig migration [8], [9]	Incremental legacy decommissioning	Traffic-routing facade, dual-run	Low (execution log only)	Domain-general; no core-banking dual-ledger reconciliation discipline
Event-driven / API-first banking / ISO 20022 real-time payments [3], [4], [15]–[17]	Real-time integration, API security, and payment-rail modernization	Event-driven microservices, zero-trust API gateway, ISO 20022 messaging	Moderate	Integration-layer and payment-standard only; not linked to resilience or AI-observability design
Resilience / SRE / chaos engineering [11], [12], [18]–[20]	Operational resilience, regulatory obligation, and availability engineering	Chaos experiments, multi-region active-active, principles-based and directly enforceable regulation	Low–Moderate	Domain-general; no blast-radius linkage to migration sequencing or AI autonomy
Agentic/GenAI in banking [5]–[7], [21]–[23]	Autonomous and generative AI adoption	LLM agents, ReAct, multi-agent orchestration	Low–Moderate	Domain-general agent literature; no linkage to core ledger action governance
XAI for credit risk and fraud [24]–[26]	Explainable risk and fraud scoring	SHAP/LIME on ensemble models	High	Assumes real-time data is already available; silent on the core architecture that must supply it
Data mesh / data fabric [27]	AI-ready data organization	Domain ownership, federated governance	Moderate	Treated independently of core decomposition and API/event contract design

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 10 October 2026
Page No: 5937 - 5957
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0763

The CORE approach is suggested as a means of filling that integration gap.

3. The problem statement and the reason for the design

The framework is motivated by three structural weaknesses. The first of these is that in most transformation programs the order in which migration activities are carried out is based on technical dependencies and on the convenience of vendor deliveries rather than on business risk and blast radius; this is exactly the pattern identified in the published analysis as being associated with programs that continue to run legacy functionality years after go-live: the most difficult and high-risk capabilities are postponed until the end of the wave plan rather than being deliberately sequenced, and by the time the program gets to them, both budget and executive attention as well as organizational willingness to tolerate further disruption have already been greatly reduced [1]. The second weakness is that the integration contracts a bank draws up during the modernization phase are generally tailored to the channels and partners that were known at the time of design, not to the observability an AI system will need many years later in order to reason safely about the resulting services — an API designed to return a current balance when requested is not the same as an event stream that enables an embedded AI system to observe each balance-affecting event as it occurs, and it is considerably more expensive to introduce the latter after a composable core is already in production than it is to have designed for it from the beginning. The third issue is that decisions regarding resilience investment and AI autonomy are almost never based on the same risk model: the multi-region failover arrangement of a service is usually determined by an infrastructure team using operational-availability criteria, while the level of autonomy granted to an embedded AI system over that same service is typically decided by a separate model-risk or AI-governance team using a completely different risk classification, so a service that in reality has a low blast radius and is highly reversible may still end up receiving costly resilience investments while a service that genuinely has a high blast radius receives an AI autonomy grant that is based solely on model accuracy and not on what happens when the model makes an error. The system is designed to meet four requirements that are a direct result of those weaknesses: (i) when planning migration wave sequencing it is necessary to use a clear and jointly agreed blast-radius and business-risk model, not merely the order of technical dependencies, and this same model must be used to guide decisions regarding resilience investments and AI autonomy for the resulting service, so that a bank ends up with a single risk approach for each capability rather than having three that are independently determined; (ii) integration contracts—whether APIs or events—must be designed with regard to the entire lifecycle of the consumers a capability is likely to have, including AI systems that will reason about live event streams, not just the channels that were known at the time of initial design; (iii) resilience engineering must be clearly traceable to the specific operational-resilience obligation (such as a tolerance-for-disruption threshold or a critical-operation designation) that it is addressing, so that the bank's resilience investments can be justified to a supervisor and not just to an internal availability target; and (iv) every autonomous or AI-assisted action taken with respect to the modernised core must include an explanation that is generated from the same evidence that the underlying model used, and this explanation must be logged in a format that allows a risk reviewer, an auditor, or a supervisory reviewer to examine it directly, so that "AI-ready" truly means governable and not just that the technology is technically capable.

4. The CORE Framework

4.1. The design principles and layered architecture

CORE is made up of five layers, as illustrated in Figure 1: (1) the legacy core, the channel, the payment rail, and third-party data and event sources; (2) the ingestion process and a common data and event fabric that covers both the legacy environment and the new composable services; (3) four parallel intelligence and execution engines — Cloud-native Modernization, Open API and Event Orchestration, Resilience and Continuity, and Embedded Intelligence; (4) a shared explainability and observability layer which presents all of the engines' decisions in a single, consistent format; and (5) a governance and human-in-the-loop review layer — comprising the change-advisory board, the model-risk function, and the operational-resilience owners — whose decisions are fed back into layers 1 to 3. Just as in other high-stakes automated-decision domains

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 10 October 2026
Page No: 5937 - 5957
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0763

that have similar layered governance structures, this arrangement ensures that each engine can make its own independent choices regarding the internal tools and techniques it uses while requiring all four of them to publish their outputs into one common schema, so that the governance layer, the regulator-facing evidence trail, and the AI systems that finally use the modernized core can all interpret the information in the same way.

Figure 1. Layered CORE Architecture

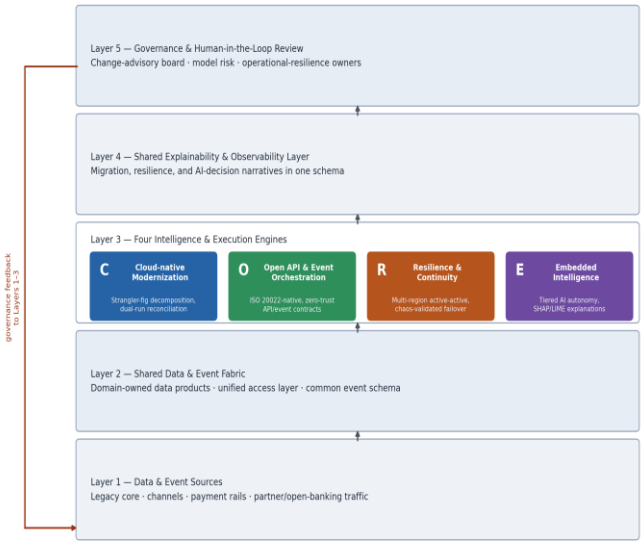

Figure 1 The layered CORE architecture

Figure 1 illustrates the layered CORE architecture, in which the legacy, channel, and payment-rail sources feed into a common data and event fabric, which in turn feeds four parallel engines (Cloud-native Modernization, Open API/Event Orchestration, Resilience and Continuity, Embedded Intelligence), a shared explainability and observability layer, and a governance layer that updates the migration wave plan, the API/event contracts, the resilience posture, and the AI autonomy policy.

4.2. The first to second layer: data and event sources, ingestion, and the shared fabric

Layer 1 includes all the sources that are actually affected by the core banking transformation program: the existing ledger and its batch extract files; the increasing number of composable services that have already been migrated as the strangler-fig wave plan moves forward; the channel systems (mobile, web, branch, contact centre); connectivity with the payment networks, including the ISO 20022-native real-time rails such as FedNow and similar instant-payment schemes in other jurisdictions [3], [4]; the open-banking and partner API traffic; and the bank's current risk, fraud, and compliance systems, which the modernised core must continue to feed without any disruption during the migration process. Layer 2 carries out three tasks with these diverse sources: it applies a domain-oriented, product-of-data approach — in accordance with the data mesh pattern of federated, domain-owned data products [27] — for each capability domain (deposits, payments, lending, cards) as the migration progresses, so that data ownership remains accountable to the team that actually has understanding of the domain rather than being handed over by default to a central data team lacking domain knowledge; it makes the federated domain data available via a single, metadata-driven access layer in the sense of a data-fabric [27], so that a consumer — whether it be a channel, a partner, or an AI system — does not need to know which of the numerous domain-owned services currently holds a particular fact, or whether that fact is still being provided by the legacy core or has already been taken over by a migrated service; and it records every state-changing event, both from the legacy system and the new ones, into a common event schema, so that the Embedded Intelligence Engine (Section 4.3.4) can observe a consistent event stream across the migration boundary rather than losing the ability to observe the system the moment a capability moves from the legacy core to a new service.

4.3. The third layer: the Four Intelligence and Execution Engines

4.3.1. Cloud-Native Modernization Engine

The engine is responsible for carrying out both the decomposition and the migration from the traditional monolith to the target composable architecture. It implements the strangler-fig pattern [8], [9] on a capability by capability basis: a routing façade is placed in front of each existing capability, traffic is gradually diverted to the newly developed, cloud-native, 5946ontainerized service equivalent, the two systems operate in a dual-write or shadow-read reconciliation mode with respect to the same official ledger until the output of the new service has been checked against that of the legacy system while it is running under real production conditions, and it is only at that point that the old pathway is officially

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 10 October 2026
Page No: 5937 - 5957
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0763

decommissioned—never being left running quietly as unused but still maintained code, since the transformation-outcome analysis that identifies this specific failure mode points out that it is a recurring cause of the ongoing presence of legacy systems [1]. The decomposition boundaries are based on the BIAN service landscape and its coreless-banking pattern [10] rather than on the internal module boundaries of the legacy system, in order that the resulting services will be able to compose properly with the vendor and in-house composable platforms mentioned in Section 2.2 [13], [14] rather than simply reproducing the coupling found in the original monolith in a new form that is merely more fashionable.

Table 2 Illustrative capability decomposition with the migration-wave and AI-autonomy tiering matrix

Balance inquiry / account statement	**Current Account Fulfillment (read path)**	**Strangler-fig, shadow-read validation**	**Low — read-only, fully reversible**	**Single-region with warm standby**	**Tier 1 — Autonomous (AI-generated statement summaries, proactive balance alerts)**
Standard deposit posting	Current Account Fulfillment (write path)	Strangler-fig, dual-write reconciliation	Moderate — reversible via correcting entry, single-account blast radius	Multi-region active-active	Tier 1 — Autonomous for pre-verified, policy-conforming transactions
Real-time payment initiation and settlement	Payment Execution (ISO 20022-native)	Parallel build, phased traffic cutover behind routing facade	Moderate–High — time-critical, reversible only via costly reversal/recall	Multi-region active-active, chaos-tested failover	Tier 2 — Approval-gated for anomalous amount/counterparty patterns; Tier 1 for pre-verified recurring payments
Credit line / overdraft decisioning	Consumer Lending Origination	Big-bang cutover after extended dual-run validation	High — direct customer financial exposure, regulatory fair-lending scrutiny	Multi-region active-active	Tier 2 — Approval-gated; AI proposes with SHAP-based explanation, human underwriter authorizes
General ledger / regulatory reporting close	Financial Accounting & Reporting	Deferred to final migration wave after all product domains stabilize	High — irreversible once regulatory filings are submitted	Multi-region active-active, extended chaos-testing window	Tier 3 — Human-led; AI drafts reconciliation narrative only, controller retains sole authority

4.3.2. Open API and Event Orchestration Engine

The engine makes all migrated capabilities available — and, via an API interface, all legacy capabilities that have not yet been migrated — accessible both as synchronous APIs and as asynchronous event streams, enabling a consumer to select the interaction model that actually meets the requirements of its particular use case rather than being obliged to use a request-response approach for a use case (for example, an AI system that needs to observe balance

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 10 October 2026
Page No: 5937 - 5957
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0763

changes as they occur or a partner carrying out real-time payment reconciliation) for which event-driven access is far more appropriate [15]. The API contracts are versioned and are based on the BIAN semantic API vocabulary [10] in order to ensure interoperability, and real-time payment connectivity is implemented from the beginning in a way that is native to ISO-20022 rather than having it translated at the boundary, since the approach of translating at the boundary has in the past been the source of the message-mapping technical debt which the modernisation of real-time payment systems is meant to eliminate [3], [4]. Since the composable, API-exposed core increases the attack surface compared to a mainframe that can only be accessed through strictly controlled internal channels, each API call that this engine routes — including traffic between internal services — is authenticated and authorized in accordance with a zero-trust policy rather than being implicitly trusted based on its network location, following the zero-trust API gateway pattern that has been specifically documented for digital banking [16], [17]. As shown in Table 2 Illustrative capability decomposition with the migration-wave and AI-autonomy tiering matrix.

4.3.3. Resilience and Continuity Engine

The resilience engine is the mechanism through which CORE turns formal operational-resilience obligations into action, rather than treating resilience as just a general availability target. When a capability is being migrated, the engine assesses its criticality classification — that is, whether it constitutes a "critical operation" in the sense in which the Basel Committee uses the term — and assigns it an explicit maximum tolerable disruption tolerance [11]. On this basis it establishes the appropriate resilience posture: for capabilities that have the highest level of criticality a multi-region active-active deployment is provided so that, in the event of a failure of a regional infrastructure, service continues from a healthy region rather than requiring a slower failover process [20], while for capabilities that are genuinely of lower criticality a correspondingly lighter resilience posture is put in place so that resilience investments are proportional rather than all being made to the maximum level. It continuously checks that this resilience posture is valid, not just at design time, by using chaos engineering to introduce controlled failures — such as regional outages, spikes in dependency latency, and loss of payment-rail connectivity — against both the newly migrated services and the existing dual-run legacy system, in accordance with the practice described in the financial-services chaos-engineering literature [18], [19], in order that any resilience gap may be found during a scheduled experiment rather than during a live incident when the migration programme is at its highest risk: that is, in the weeks immediately after a capability's cutover, before the new service has built up a significant production track record.

4.3.4. Embedded Intelligence Engine

It is the fourth engine that enables the updated core to be truly AI-ready rather than just AI-compatible. By drawing on the shared event fabric (Section 4.2), it provides embedded AI systems — such as a fraud-detection model, a 5948ersonalization engine, and an agentic servicing assistant — with real-time, well-governed access to ledger and payment activity as it occurs, and it ensures that every action proposed by the AI is mediated through the same autonomy-tiering framework that has been developed in the literature on agentic systems for other high-stakes operational areas: the action either proceeds autonomously, is sent for approval, or is escalated to a human reviewer, with the level of control being tailored specifically to the nature of the action in terms of its reversibility and blast radius rather than using a single autonomy setting that is applied uniformly throughout the entire scope of the AI system [5]–[7], [22], [23]. Table 2 shows an example of how this tiering is combined with migration-wave sequencing, since — in line with the design rationale given in Section 3 — the CORE system uses the classification of a capability’s blast radius as the sole input that determines its migration risk tier, its resilience posture, and its AI autonomy tier all at the same time, rather than calculating these three aspects separately. The risk-assessment and decision-making models incorporated through this engine — for example, credit-risk assessment and fraud detection — must also provide the same kind of local attribution and counterfactual explanation that the XAI-for-banking literature has shown is possible without having to give up predictive performance [24]–[26], so that when an application is rejected or a transaction is blocked, the customer can be given an explanation and the situation can be defended to an auditor from the very first time the system goes live,

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 10 October 2026
Page No: 5937 - 5957
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0763

not only after the requirement has been raised during a subsequent model-risk review.

As shown in Table 2 presents a decomposition of illustrative capability, together with a migration-wave and an AI-autonomy tiering matrix, by means of which CORE determines the migration risk tier, the resilience posture, and the AI autonomy tier from a common blast-radius classification.

Figure 2. CORE Decision Loop

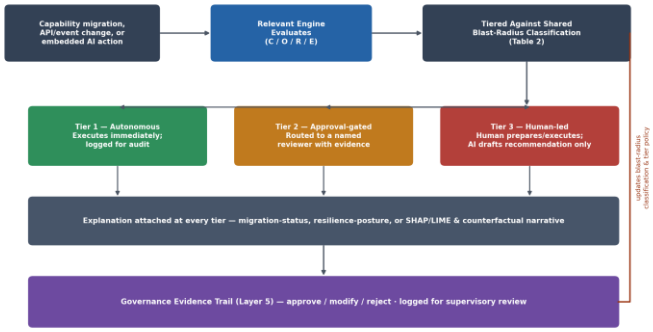

Figure 2 In the CORE decision loop, a capability migration, an API/event contract change, a resilience posture decision, or an embedded AI action is assessed, compared to the shared blast-radius classification given in Table 2, and sent to autonomous execution, approval, or human-led handling, with an explanation provided at each stage and the decisions recorded in the governance evidence trail

4.4. The fourth layer: the shared explainability and observability layer

All engines make their output available — including the status of the Modernization Engine's cutover and the results of its dual-run reconciliation, the version numbers of the API/event contracts and the traffic-routing status of the Orchestration Engine, the chaos-test results and failover position of the Resilience Engine, and the model decisions together with their explanations from the Embedded Intelligence Engine — in a common schema, enabling a program-level dashboard to display a single, unified view of the progress of the transformation and the associated risks rather than four separate and disconnected status reports. The explainability layer consistently presents three types of narrative: a migration-status narrative which clearly states which capabilities have cut over, which are still running in dual mode, and which remain completely on the legacy path; a resilience-posture narrative which indicates the specific critical-operation classification and the tolerance-for-disruption threshold that the resilience investment for a given capability is meeting; and, for each Tier 2 or Tier 3 AI decision, an explanation narrative — based on SHAP/LIME attribution or a counterfactual statement — that names the particular factors which led to the model's output, so that when a credit or fraud decision is reviewed by a human it is already accompanied by an explanation rather than just being presented as a raw score. Figure 2 illustrates how a single transformation or AI decision passes through the appropriate engine and is then directed, with its explanation attached, to the governance surface that the situation requires.

4.5. The fifth layer: governance, orchestration, and feedback

A governance dashboard — covering the transformation program's change-advisory board, the bank's model-risk and AI-governance functions, and the operational-resilience owners that the Basel Committee requires a bank to designate [11] — displays the open migration cutover decisions, the results of resilience tests, and the Tier 2 and Tier 3 AI cases to the appropriate reviewer group, together with full context and explanations, enables the reviewers to approve, modify or reject the dispositions and keeps a record of each disposition in an evidence trail that a supervisor can examine directly. Instances in which a cutover is confirmed as having been successful and AI decisions are correctly tiered reinforce the blast-radius classification for that capability class; those in which decisions are disputed or reversed are included as input for the following tuning cycle both for the migration wave plan and for the autonomy tiering model; and any capability whose migration is continually deferred or whose AI autonomy tier is repeatedly overridden by reviewers is shown as a program-level risk item instead of remaining as a standing exception — this being the means by which CORE is designed to bring the program towards complete, well-governed migration rather than allow the kind of persistent, un-decommissioned legacy footprint that the transformation-outcome analysis identifies as the normal rather than the exception [1].

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 10 October 2026
Page No: 5937 - 5957
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0763

5. A practical example of a study involving a composable deposits-and-payments core in the face of a surge in real-time payment demand

In order to make the framework more specific, this section considers a hypothetical yet typical situation (as shown in Figure 3), involving a super-regional bank in the middle of a multi-year core banking transformation – a kind of scenario that an architect actually running such a project at this level would be familiar with. At T0 the bank has already transferred its current-account read and standard-deposit-posting functions from the legacy mainframe to a composable, cloud-native service as part of the Cloud-native Modernization Engine's strangler-fig approach, and is at the stage of migrating real-time payment initiation, which is still being processed in parallel in dual-write mode alongside both the existing batch payment engine and the new ISO 20022-native payment service, in accordance with the Tier 2 classification in Table 2. At T1 the bank begins to offer its real-time payment service to a group of high-value commercial customers and, during the first week of trading, the volume of transactions using the new payment service increases significantly as these customers move their activity from batch ACH to the new instant payment rail. The real-time payment traffic handled by the Open API and the Event Orchestration Engine reaches a volume that had not been anticipated in the original capacity plan at this stage of the migration and starts to queue instead of processing the settlement instructions within the settlement window required by the payment rail. At T2 the Resilience and Continuity Engine's monitoring — having been tested in a chaos-engineering exercise targeting this exact failure mode three weeks earlier — picks up the queuing before it becomes visible to customers, and because of the engine's multi-region active-active setup for this Tier 2 system with a moderate-high blast-radius capability it is able, within the time period permitted by the rail's settlement SLA, to redirect a part of the surging traffic to the payment processing capacity of a healthy secondary region while at the same time the primary region increases its own capacity. At T3, the Embedded Intelligence Engine notices the increased transaction volume being transmitted via the shared event fabric and identifies a small group of transactions in that surge which differ from the established behavioural baseline of the flagged commercial customers in both the amount and the counterparty — this is in accordance with the Tier 2 approval-gating rule listed in Table 2 regarding anomalous amount/counterparty patterns for real-time payments. Instead of blocking the transactions immediately, since that could lead to a false positive and a service disruption at a time when transaction volumes are already high, the engine sends them over for accelerated review by a fraud analyst, together with a SHAP-based explanation that names the specific features of behavioural deviation which caused the transaction to be flagged, while at the same time allowing the rest of the volume, which conforms to the expected pattern, to continue being processed automatically according to the Tier 1 rule for pre-verified recurring payment activity. At T4, the fraud analyst confirms that the group of transactions in question is legitimate — the customer in question had, in fact, changed its payment counterparty as part of a normal vendor transition — and does so within minutes, using the explanation attached rather than having to work out the anomaly from the raw transaction logs. At T5, the surge dies down, the dual-write reconciliation between the old batch payment engine and the new service confirms that the two ledgers stay in agreement during the entire incident, and the case — comprising the resilience engine's regional traffic shift, the embedded intelligence engine's flag and clearance, and the confirmed reconciliation — is kept as governance evidence and then fed back into Layer 5, both to readjust the original capacity model which had underestimated the surge and to verify that the Tier 2 autonomy classification functioned as intended.

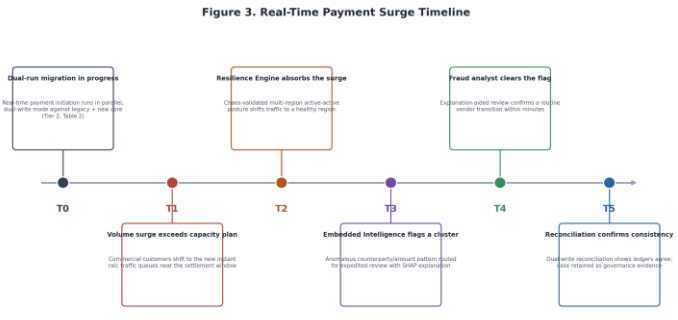

Figure 3 Timeline illustrating a surge in real-time payments during a dual-run migration (T0), when a volume surge goes beyond the original capacity plan (T1), the resilience engine's chaos-validated multi-region setup deals with it (T2),

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 10 October 2026
Page No: 5937 - 5957
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0763

the embedded intelligence engine identifies a cluster as anomalous and triggers urgent human review together with an explanation (T3), the fraud analyst removes the flag after examining the explanation (T4), and reconciliation then verifies consistency of the ledger, closing the case with a full evidence trail (T5)

This example shows that the four engines have to function as a single, unified system rather than as four separate programs. On its own, the Modernization Engine's dual-write reconciliation could have only detected a discrepancy in the ledger after the fact, which was too late to avoid the delay in settlement; without the Resilience Engine's pre-provisioned, chaos-validated capacity, the Orchestration Engine's traffic routing would not have had any spare regional capacity to redirect the traffic to; and by itself, the Resilience Engine's regional shift would have processed the group of anomalous transactions without identifying them, since the raw traffic volume alone was not the indicator that mattered. It is only through the four engines sharing the same blast-radius classification and the same event fabric — that is, by all of them simultaneously observing the same surge in real time from four different positions — that a mistake in capacity planning is turned into an absorbed surge and a correctly prioritized anomaly rather than either a customer-visible outage or an unreviewed, possibly fraudulent payment.

6. Methodology

6.1. The sources of the data and the process of feature engineering

In accordance with Section 4.2, feature engineering makes use of three levels of data that include both the existing and the modernized systems: (i) structural features that describe each capability (such as the BIAN domain classification, the blast-radius tier from Table 2, and the dependency depth on other capabilities that have not yet been migrated); (ii) operational and behavioural features which are aggregated by capability and by customer over defined time periods (including the transaction volume compared to the historical baseline, the rate of discrepancy between dual-run reconciliations, the pass/fail history from chaos tests, and the resilience-posture utilisation during the most recent regional shift); and (iii) AI-decision-trace features obtained from the Embedded Intelligence Engine's own execution log (indicating which model made a particular decision, which features were used to attribute that decision, and whether the decision was confirmed, overridden, or escalated by a human reviewer). Each of the four engines receives input from all three tiers, the Resilience and Embedded Intelligence engines placing the greatest weight on the second and third tiers since the operational behaviour and decision-trace evidence provide the most direct indication of whether a capability's current risk classification still aligns with its actual production behaviour.

6.2. Model and Technique Selection

Choosing the right model and method is key when working with data. Start by looking at the problem, the type of data you have, and the goal. For example, use classification methods for sorting items into categories. Use regression methods for predicting numbers. If you work with pictures or speech, try convolutional neural networks or recurrent neural networks. You also need to think about how much data you have and how complex the problem is. Simple models like linear regression or decision trees often work well when you do not have much data or when the relationships are easy to spot. More complex models can catch complicated patterns but need more data and computer power. Test many models and compare their results. Use tools such as cross-validation to see how well each model does. Pick the model that gives the best results for your problem. The dual-run reconciliation feature of the Cloud-native Modernization Engine is intentionally based on a deterministic comparison procedure — that is, it compares the output of the legacy system with that of the new service for the same transaction — rather than relying on a learned model, since the whole point of the reconciliation stage is to establish bit-for-bit equivalence (or, in cases where legitimate rounding or timing differences occur, equivalence within a defined tolerance) before the capability is allowed to switch over [8], [9]. Just as the contract versioning and zero-trust authorization checks of the Open API and Event Orchestration Engine are rule-based and deterministic, so too is this in line with the zero-trust API gateway approach used in digital banking [16], [17], because an authorization decision taken at the API gateway must be just as immediately traceable to the policy that generated it as the guardrail decisions outlined in the adaptive-security-governance

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 10 October 2026
Page No: 5937 - 5957
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0763

literature. The Resilience and Continuity Engine combines the results of chaos-engineering experiments [18], [19] with statistical capacity and failure-mode modelling in order to define and verify each capability's resilience position, in accordance with the multi-region active-active deployment approach described for high-availability banking systems [20]. The risk-scoring models employed by the Embedded Intelligence Engine — namely those for credit decisioning and fraud detection — use gradient-boosted ensembles together with SHAP/LIME attribution, in keeping with the proven combination of strong predictive performance and individualised explainability found in the credit-risk and fraud-detection XAI literature [24]–[26], while its agentic servicing and personalisation components adhere to the ReAct-style interleaved reasoning-and-acting pattern [22], being composed of coordinating specialist agents for tasks covering multiple capability domains rather than relying on a single generalist agent trying to handle all domains [23].

Table 3 Evaluation metrics per engine and comparable benchmark ranges reported in the literature and industry reporting (calibration targets for pilot validation, not results from a live CORE deployment)

Engine	Metric(s)	Comparable Benchmark in Literature / Industry Reporting	Source
Cloud-native Modernization	Full migration success rate; dual-run reconciliation discrepancy rate	Industry analysis reports ~30% full-migration success rate and 10–20% residual legacy footprint as the current norm for core banking transformations generally	[1]
Open API and Event Orchestration	Contract stability; zero-trust authorization latency; API/event coverage of migrated capabilities	Zero-trust API gateway designs for digital banking report authorization enforceable at per-call granularity without materially degrading transaction latency	[16], [17]
Resilience and Continuity	Chaos-test coverage; failover/regional-shift latency vs. designated tolerance-for-disruption	Chaos-engineering practice and multi-region active-active designs for financial services report growing tool maturity and sub-minute regional-shift capability in comparable architectures	[18], [19], [20]
Embedded Intelligence	Precision/Recall/F1/AUC-ROC for risk scoring; reviewer-confirmed tiering precision; explanation usefulness	XAI-augmented credit-risk and fraud-detection models report strong predictive performance alongside individualized, auditable local explanations in comparable settings	[24]–[26]
Shared Explainability and Governance Layer	Reviewer-rated explanation usefulness; time-to-disposition for Tier 2/3 cases	Agent- and AI-governance literature identifies genuine, execution-path-integrated human oversight as the differentiator for reviewer trust in comparable adaptive-risk-governance domains	[7], [21], [23]

6.3. Explainability Techniques

The same three types of explanation are employed by all four engines, reflecting the narrative forms described in Section 4.4: migration-status narratives, which are produced directly from the dual-run reconciliation trace and specify precisely which capability, what traffic percentage, and which validation criterion underlies a cutover decision; resilience-posture narratives that name the particular

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 10 October 2026
Page No: 5937 - 5957
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0763

critical-operation classification and the chaos-test evidence which justify a specific resilience investment; and SHAP/LIME feature-attribution and counterfactual narratives used by the Embedded Intelligence Engine in making its risk-scoring and anomaly-flagging decisions, these being generated from the same features that the underlying model itself used [24]–[26]. The approach taken—which is by design traceable—consists of producing the explanations from the same execution trace and reconciliation evidence that the engines generated, rather than treating the explanation as a separate post-hoc interpretation; this is because both the bank's internal audit function and its prudential supervisor need to examine a decision in light of the platform's own record of what actually occurred, not against an approximation of what happened.

6.4. The plan for evaluation and the metrics

Since this manuscript offers a reference architecture and an example of a case study rather than presenting results from a specific production transformation, the evaluation outlined here is to be carried out as a plan in relation to a pilot transformation programme, with target ranges based on the closest comparable benchmarks that have been published. Table 3 gives a summary of the metrics for each engine and of the benchmark ranges cited in the literature and in industry case studies for similar techniques; these figures serve as calibration targets for validating the pilot, not as results actually measured by CORE. The Modernization Engine is assessed according to the migration success rate (that is, complete cutover and the decommissioning of the legacy systems, not partial migration) and on the rate of discrepancies found during dual-run reconciliation, with these being benchmarked against the approximately 30% full-migration success rate that is found in industry analysis reports concerning core banking transformations as a whole [1]; the Orchestration Engine is assessed on API/event contract stability and on zero-trust authorization latency; the Resilience Engine is assessed on chaos-test coverage (that is, the proportion of the provisioned capabilities which have a current passing chaos-test result) and on the actual failover or regional-shift latency in relation to the capability's stated tolerance-for-disruption threshold [11], [20]; and the Embedded Intelligence Engine is assessed using the standard classification metrics (precision, recall, F1, AUC-ROC) for its risk-scoring models, on the precision of its autonomy-routing decisions as confirmed by reviewers, and on the usefulness of its explanations as rated by reviewers and collected through structured feedback in the Layer-5 governance dashboard.

7. Things to consider in terms of implementation and deployment

7.1. Integration with the existing core system and the vendor's estate

CORE is intended to be incorporated inside a bank's current transformation initiative rather than being placed alongside it: it takes as input the bank's existing legacy core system, whatever composable or vendor core banking platform the bank has chosen as its target (such as Temenos, Mambu, or an equivalent composable platform [13], [14]), and the bank's current risk, fraud, and compliance tools, and is developed as a governance and orchestration layer to coordinate these systems rather than as a core banking product on its own. This approach to integration is in line with a practical situation that is well known to any 5953rganization carrying out a core banking transformation, since the vendor selection, the decades of business logic accumulated by the legacy platform, and the bank's existing risk and compliance tools all involve commitments that are either already in place or well advanced when an architectural framework such as CORE is adopted; the most efficient and low-risk way to achieve a genuinely integrated programme is therefore to co-ordinate these already-existing commitments rather than requiring the bank to re-examine them.

7.2. The deployment architecture, the sequencing of the migration wave, and the latency budget

The four engines of CORE have considerably different requirements when it comes to latency and sequencing, and the deployment architecture as well as the migration wave plan must specifically take these into account. The Modernization Engine can accept near-real-time latency rather than synchronous latency since its function is one of validation and not actual transaction processing; the Orchestration Engine's API and event routing must meet the latency budget that a particular channel or payment rail actually demands, this end-to-end latency for real-time payments being measured in a few seconds [3], [4]; the Resilience Engine's decisions regarding a regional shift must be made quickly enough to remain within the level of disruption tolerance that the Basel

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 10 October 2026
Page No: 5937 - 5957
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0763

Committee requires banks to have explicitly defined for each critical operation [11]; and the Embedded Intelligence Engine's Tier 1 autonomous decisions must be made within the same latency budget as the underlying transaction, while Tier 2 and Tier 3 decisions can allow for the longer latency that is introduced by a human review stage. The sequencing of the migration waves is a direct result of the shared blast-radius classification in Table 2 and not simply because of the order of technical dependencies: the capabilities with a lower blast radius and high learning value (such as read-path balance inquiry and standard deposit posting) are given priority first, both because of their lower risk level which is appropriate for a program that is still developing its delivery capabilities and because the reconciliation and resilience-validation discipline established during those early waves is what reduces the risk associated with the higher-blast-radius capabilities (such as real-time payment execution, lending decisioning and the general ledger close) that are scheduled towards the end of the program — directly addressing the specific failure pattern, which the published transformation-outcome analysis identifies as the main cause of the persistent legacy footprint, of delaying the most difficult capabilities without a deliberate plan to eventually migrate them [1].

7.3. Change Management, Regulatory Engagement, and Adoption

The fact that CORE's actions target a bank's system of record and, in the case of its most advanced capabilities, specifically designated critical operations means that rolling out the system is as much a change-management and regulatory engagement exercise as it is an engineering one: the migration of each capability, its resilience position, and its AI autonomy level must be examined by the bank's operational-resilience and model-risk governance functions before the cutover takes place, not afterwards, and — in line with the emphasis in the published transformation-outcome analysis on having dedicated delivery squads rather than delivery led by the operations team [1] — the program should be set up and managed as a permanent transformation capability with its own accountable leadership, not be regarded as a temporary project that disappears when the old platform is switched off. It is this step-by-step trust rollout approach — testing each capability's AI autonomy level initially in an advisory capacity before advancing it to its intended level — that ensures CORE's tiering model is firmly based on the bank's actual risk tolerance and level of operational maturity, rather than on the illustrative tier assignments listed in Table 2, which should be seen as a basis for adaptation rather than as a fixed policy.

8. Limitations

The evaluation ranges proposed in Table 3 are not derived from a specific, named production deployment, but instead based on core banking transformation practice and published industry analysis; similarly, the reference architecture and illustrative case study are presented in a non-specific manner. The blast-radius tiering matrix provided in Table 2 is intended to be illustrative and a starting point for determining tiering for any specific capability; other banks with varying product mixes, regulatory footprints, and legacy platform characteristics may reasonably be expected to tier the same nominal capability differently. The framework also presumes a degree of executive sponsorship and delivery capacity which published transformation-outcome analysis shows is often missing in practice [1]; CORE's architecture can minimize the technical risk a transformation programme faces, but it cannot compensate for the organisational commitment that the same analysis reveals is the more fundamental component of successful transformations. Finally, the case study reported in Section 5 is synthetic and illustrative – it's designed to show the way the framework operates, and the values for timing, volume and thresholds are for illustrative purposes only, and not necessarily validated.

Conclusion and Future Work

For most large banks, core banking transformation has been a technology replacement project and has a poor, well-documented delivery track record. What this manuscript has posited is that the pattern underlying that track record is architectural, rather than organizational: modernization writeups, integration writeups, resilience writeups and AI readiness writeups are written as distinct writeups and deployed as distinct projects, and the resulting misalignment — a migration wave plan that is sequenced without regard to blast radius, an API contract that doesn't support the AI observability that a bank will require years later, a resilience posture calibrated to one set of criteria for the same capability as an AI autonomy grant submitted for another — is

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 10 October 2026
Page No: 5937 - 5957
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0763

the root cause of stalled migrations, unrealized performance benefits, and AI projects that can't safely access the data and actions they need. Here, CORE's contribution is architectural: how cloud-native modernization, open API and event orchestration, resilience and continuity and embedded intelligence can share a blast-radius classification, a data and event fabric, and an explainability layer, and a bank can design a single risk posture per capability without having to reconcile four different risk postures ex post. The payoff is illustrated in Section 5: When a capacity plan didn't model a real-time payment demand surge, all five engines read the same event stream through the same shared risk classification lens, instead of four disconnected lenses, which means the surge is absorbed without any disruption to the customer. Further, an anomalous transaction cluster in that surge is correctly triaged, neither outright blocked or simply missed. There is three priority for future work. Firstly, empirical validation of the framework against a pilot core banking transformation program using a production or de-identified transformation dataset to update the ranges from literature and industry analysis in Table 3 with CORE's own measurements, based on a real core banking transformation wave plan, resilience-test history and AI-decision population. The second is to make the blast-radius classification introduced in Section 4.3.4 more systematic, based on measured reconciliation discrepancy rates, measured chaos-test results, and measured AI-decision override rates, instead of the illustrative, expert-assigned classifications in Table 2. The third is a longitudinal study of the impact of CORE's shared blast-radius model on completion rates specifically: does a program done under CORE's shared blast-radius model measurably close the gap between the roughly 30% success rate in completing full migration found in published analysis reports and a true, well-governed completion, which is what a true intelligent core banking transformation should require, as modernization, integration, resilience, and AI-readiness are one, rather than four coordinated afterthoughts?

References

[1]. McKinsey & Company, 'How to get a core banking transformation right: Eight mistakes to avoid', Tech Forward insights. Online at: https://www.mckinsey.com/capabilities/tech-and-ai/our-insights/tech-forward/how-to-get-a-core-banking-transformation-right-eight-mistakes-to-avoid

[2]. DXC Technology, in DXC Insights, asks 'Why do banks still rely on cobol-driven mainframe systems?' Available at https://dxc.com/insights/knowledge-base/blogs/why-banks-still-rely-on-cobol-driven-mainframe-systems

[3]. Federal Reserve Financial Services, “The FedNow Service and ISO 20022” . Available at https://www.frbservices.org/financial-services/fednow/what-is-iso-20022-why-does-it-matter

[4]. The Federal Reserve Bank of Minneapolis, 'ISO 20022 the standard at the heart of payments system modernization', 2021. Available at https://www.minneapolisfed.org/article/2021/iso-20022-the-standards-at-the-heart-of-payment-system-modernization

[5]. Gartner, ‘2026 Hype Cycle for Agentic AI’. Available at https://www.gartner.com/en/articles/hype-cycle-for-agentic-ai

[6]. Gartner, 'Emerging Agentic AI Technologies in Banking and Investment Services That All CIOs Must Prepare For'. Available at https://www.gartner.com/en/documents/7677761

[7]. McKinsey & Company, 'State of AI trust in 2026: Shifting to the agentic era', Tech Forward insights. Online at: https://www.mckinsey.com/capabilities/tech-and-ai/our-insights/tech-forward/state-of-ai-trust-in-2026-shifting-to-the-agentic-era

[8]. Amazon Web Services, in its Prescriptive Guidance, describes the 'Strangler fig' pattern. Available at https://docs.aws.amazon.com/prescriptive-guidance/latest/cloud-design-patterns/strangler-fig.html

[9]. ResearchGate preprint titled "Microservice Migration Using Strangler Fig Pattern: A Case Study on the Green Button System". Available at https://www.researchgate.net/publication/349568403_Microservice_Migration_Using_Stra

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 10 October 2026
Page No: 5937 - 5957
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0763

ngler_Fig_Pattern_A_Case_Study_on_the_Green_Button_System

[10]. The Banking Industry Architecture Network (BIAN), 'Coreless Banking'. Available at https://bian.org/deliverables/coreless-banking/

[11]. The Basel Committee on Banking Supervision, 'Principles for operational resilience', Bank for International Settlements, March 2021. Available at https://www.bis.org/bcbs/publ/d516.htm

[12]. European Insurance and Occupational Pensions Authority, 'Digital Operational Resilience Act (DORA)'. Available at https://www.eiopa.europa.eu/digital-operational-resilience-act-dora_en

[13]. Temenos, 'Temenos Core Banking | Modular, Scalable Banking Platform'. Available at https://www.temenos.com/products/core-banking/.

[14]. Mambu offers a composable core banking platform. More information is available at https://mambu.com/en/composable-banking.

[15]. S. Padmanabham, "Building resilient Banking platforms using Event-Driven Microservices and cloud Native Architecture," International Journal of Research and Applied Innovations, volume 6, number 4, 2023. Available at https://ijrai.org/index.php/ijrai/article/view/683.

[16]. "Designing a Zero Trust Architecture to secure API Gateways in digital banking systems", Journal of Information Systems and Informatics. Online at: https://journal-isi.org/index.php/isi/article/view/1219

[17]. Zero Trust Security Model Implementation in Microservices, arXiv:2511.04925. Available at https://arxiv.org/pdf/2511.04925.

[18]. J. Owotogbe, I. Kumara, D. Di Nucci, D. A. Tamburri, and W.-J. van den Heuvel, "Chaos Engineering in the Wild: Findings from GitHub," arXiv:2505.13654, 2025. Online at: https://arxiv.org/pdf/2505.13654

[19]. Gremlin, 'Improving the reliability of financial services with Chaos Engineering'. Available at https://www.gremlin.com/community/tutorials/improving-the-reliability-of-financial-services-with-chaos-engineering

[20]. ResearchGate preprint titled "Designing Multi-Region Active-Active Cloud Architectures for High-Availability Banking Systems". Available at https://www.researchgate.net/publication/400022534_Designing_MultiRegion_Active-Active_Cloud_Architectures_for_High-Availability_Banking_Systems

[21]. DBS Bank, 'Responsible AI in banking: Gaining a competitive edge'. Available at https://www.dbs.com/artificial-intelligence-machine-learning/artificial intelligence/responsible-ai-in-banking gaining-a-competitive-edge.html

[22]. S. Yao et al., "ReAct: Synergizing Reasoning and Acting in Language Models," in Proc. ICLR, 2023; arXiv:2210.03629. Available at https://arxiv.org/abs/2210.03629.

[23]. P. Drammeh, "Multi-Agent LLM Orchestration Achieves Deterministic, High-Quality Decision Support for Incident Response", arXiv:2511.15755, 2025. Online at https://arxiv.org/abs/2511.15755.

[24]. Shreya and H. Pathak, "Explainable Artificial Intelligence Credit Risk Assessment using Machine Learning", arXiv:2506.19383, 2025. Online at https://arxiv.org/abs/2506.19383.

[25]. "Explainable AI in Big Data Fraud Detection," arXiv:2512.16037, 2025. Available at https://arxiv.org/abs/2512.16037

[26]. arXiv:2505.10050, 2025, "Financial Fraud Detection Using Explainable AI and Stacking Ensemble Methods". Available at https://arxiv.org/html/2505.10050v1.

[27]. Alation, "Data Fabric vs. Data Mesh: 2026 Guide to Modern Data." Available: https://www.alation.com/blog/data-mesh-vs-data-fabric/

[28]. NIST, "The NIST Definition of Cloud Computing", NIST Special Publication 800-145. Available at https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf

[29]. Federal Financial Institutions Examination Council, 'Joint Statement: Security in a Cloud Computing Environment', April 2020. Available at https://www.ffiec.gov/sites/default/files/m

International Research Journal on Advanced Engineering Hub (IRJAEH)
e ISSN: 2584-2137
Vol. 04 Issue: 10 October 2026
Page No: 5937 - 5957
https://irjaeh.com
https://doi.org/10.47392/IRJAEH.2026.0763

dia/press-releases/2020/2020-april-30 cloud-computing-statement.pdf