Core Banking: The Legacy Paradox

The most critical financial infrastructure in the world is held together by COBOL, mainframes, and engineers who are about to retire

Digital Finance

Why Does Your Bank Run on Software Written Before You Were Born?

The Legacy Paradox

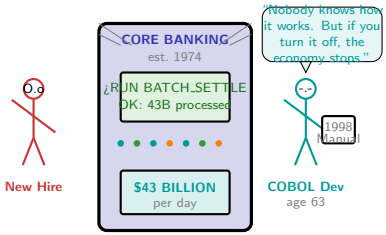
The IBM mainframes running core banking today use COBOL – a language designed in 1959. These systems process trillions of dollars every day with 99.999% uptime. They are the most reliable software on earth.

Why they still run:

- 95% of ATM transactions touch COBOL code
- 80% of in-person banking transactions run on mainframes
- Rewriting would cost billions and risk catastrophic failure
- No modern system has matched their transaction throughput

Why they are a ticking clock:

- 80% of active COBOL developers are over 60 years old
- Designed for overnight batch processing, not real-time
- Documentation is incomplete or lost entirely
- Each year of delay makes migration harder and riskier



The manual was last updated in 1998.

COBOL processes an estimated 95% of ATM transactions and 80% of in-person transactions globally – yet the average developer is over 60 and universities stopped teaching it.

What Invisible Infrastructure Sits Between Your Paycheck and Your Bank Balance?

Reflection Prompt

Your employer pays you on the 25th. You see the balance update in your banking app. Simple? Count the systems your salary actually touches between your employer's "pay" button and your available balance.

Your salary passes through at least six infrastructure layers before it arrives:

1. Employer's **payroll system** generates a SEPA (Single Euro Payments Area, EU 2008 launch) / SIC (Swiss Interbank Clearing) payment file
2. Employer's bank **core banking system** validates and queues the instruction
3. **Clearing network** (SIC in Switzerland, TARGET2 = Trans-European Automated Real-time Gross Settlement Express Transfer 2, ECB 2007, in the eurozone) nets and routes the payment to your bank
4. **RTGS (Real-Time Gross Settlement) settlement** at the central bank transfers reserves between banks
5. Your bank's **core banking system** credits your account in the nightly batch
6. Your **mobile app** reads the updated balance from a middleware layer

Each layer runs on different technology, operated by a different institution, governed by different rules. If any single layer fails, your salary does not arrive – and you may not know why for days.

Exercise: How many institutions does your salary touch? How long does each step take? What happens if the clearing network goes down on payday? Bring your answers to class.

A single salary payment crosses 6 infrastructure layers, 3–4 institutions, and 2–3 legal jurisdictions – all invisible to the person waiting for the money to appear.

What Five Systems Form the Backbone of Global Finance?

System	Function	Example	Scale
Core Banking	The brain: ledger, accounts, products, transactions	Temenos, FIS, Avaloq	Every bank
SWIFT	Messaging: standardized instructions between banks	11,000+ banks in 200+ countries	44M msg/day
RTGS	Settlement: final transfer of central bank reserves	Fedwire, TARGET2, SIC	CHF 270B/day (SIC)
CSD	Custody: who owns which securities	SIX SIS, Euroclear, DTCC	Trillions in assets
CCP	Clearing: who owes what, netting, risk management	LCH, Eurex Clearing	Reduces gross to net

The sequential chain

These five systems operate in a strict sequence for every financial transaction. A stock trade, for example, flows:

1. Your bank's **core banking** system records the order
2. **SWIFT** messages route instructions to counterparties
3. The **CCP** nets obligations and manages counterparty risk
4. The **CSD** transfers ownership of the securities
5. **RTGS** settles the cash leg with finality

Break any link and the chain stops. Each system is a single point of failure for the entire financial system.

Why this matters

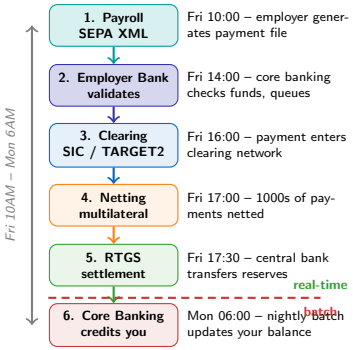
- **Core Banking** is proprietary and vendor-locked. Migration takes 3–7 years and costs hundreds of millions. Most banks run systems installed 20–40 years ago.
- **SWIFT** is a cooperative monopoly. No realistic alternative exists for cross-border messaging. Russia's exclusion in 2022 proved its geopolitical power.
- **RTGS** is operated by central banks. Settlement is final and irrevocable – the only true "cash" in the digital system.
- **CSD** and **CCP** are invisible to retail investors but critical. If a CCP fails, counterparty risk cascades through the entire market.

Key insight: Modern finance looks digital on the surface (apps, APIs, instant payments) but runs on infrastructure designed for batch processing in the 1970s.

Vendor/operator anchors: Temenos (Geneva 1993, SIX-listed); FIS (Jacksonville FL 1968); Avaloq (Zurich 1985, NEC-owned 2020); SWIFT = Society for Worldwide Interbank Financial Telecom (La Hulpe Belgium 1973, cooperative); Fedwire (US Fed, 1918); TARGET2 (Eurosystem 2007); SIC (SIX + SNB, 1987); SIX SIS (Zurich 2008); Euroclear (Brussels 1968); DTCC (New York 1999); LCH (London 1888, LSE-owned); Eurex Clearing (Frankfurt 2004); CSD = Central Securities Depository; CCP = Central Counterparty.

Five systems form the backbone: Core Banking (brain), SWIFT (messaging), RTGS (settlement), CSD (custody), CCP (clearing). Each is decades old, deeply interconnected, and nearly impossible to replace.

Follow Your Salary From Your Employer's Account to Your Bank Balance



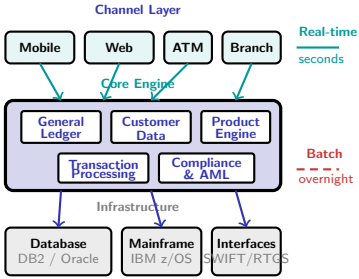
Anatomy of a salary payment

- **SEPA XML** is the standard payment format across Europe. Your employer's payroll system generates one file containing all employee payments.
- **SIC** (Swiss Interbank Clearing) processes CHF 270 billion per day. It is Switzerland's domestic clearing system, operated by SIX.
- **Netting** reduces thousands of bilateral obligations to a few net transfers. Without it, banks would need far more liquidity.
- **RTGS settlement** is final and irrevocable. Once the central bank debits one reserve account and credits another, the payment cannot be reversed.
- **Nightly batch** is why your salary often appears Monday morning even if your employer paid Friday. The core banking system updates balances overnight, not in real-time.

The bottleneck: Steps 1–5 can complete in hours. Step 6 waits for the overnight batch window. Your money sits in limbo because the core system was designed in 1974 for end-of-day processing.

SIC processes CHF 270 billion per day. Your salary crosses 6 layers – but the last step (core banking batch) is often the slowest because the system was designed for overnight processing.

What Does a Core Banking System Actually Look Like Inside?



Three layers, one bottleneck

- **Channel Layer** is modern: mobile apps, responsive web, APIs. This is what customers see. It creates the illusion of real-time banking.
- **Core Engine** is the real system: the general ledger (every debit and credit), customer records, product rules, and transaction processing. Most of this runs COBOL on IBM mainframes.
- **Infrastructure** is the foundation: DB2 (IBM relational database, first released 1983) databases from the 1980s, IBM z/OS (mainframe OS, 1964 OS/360 lineage, current z/OS since 2000) mainframes, and proprietary interfaces to SWIFT and RTGS networks.

Batch vs. real-time:

- The channel layer is real-time (you see your balance instantly)
- But the core engine updates the actual ledger in **nightly batch runs**. Your "real-time" balance is a cached approximation
- This is why transfers between banks can take 1–3 business days even though the app shows "sent"

Modern banking apps are a thin real-time layer on top of batch-processing core systems designed in the 1970s. The gap between what customers see and what actually happens is the core banking paradox.

What Happens When Fifty-Year-Old Infrastructure Finally Breaks?

The three failure modes

1. Migration disaster – TSB Bank (UK, April 2018; Sabadell-owned since 2015)

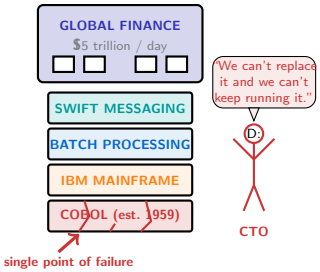
- Migrated 5.2 million customers from Lloyds Banking Group (UK) platform to Proteo4UK (Sabadell's Spanish core) over a single weekend
- 1.9 million customers locked out of their accounts
- Some customers could see other people's balances
- Cost: GBP 330 million in direct losses; CEO Paul Pester fired

2. Cyberattack – Bangladesh Bank SWIFT heist (February 2016)

- Hackers (later linked to Lazarus Group, North Korea) compromised Bangladesh Bank's (Dhaka) SWIFT terminal via the NY Fed correspondent
- Sent fraudulent transfer orders for $951 million to RCBC Manila
- $81 million successfully stolen before a Deutsche Bank typo ("Jupiter" flagged as sanctioned) triggered suspicion
- Infrastructure was unpatched, running on secondhand switches

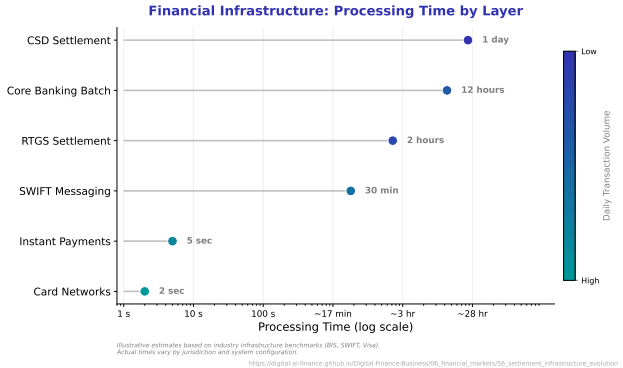
3. Knowledge extinction – COBOL (Common Business-Oriented Language, 1959, Hopper-influenced) crisis

- Average COBOL developer age: 60+
- US COVID unemployment systems crashed April 2020 – NJ Governor Murphy publicly appealed for COBOL programmers to fix them
- Universities stopped teaching COBOL decades ago



TSB lost GBP 330M migrating core banking. Bangladesh lost $81M to a SWIFT hack. US unemployment systems crashed in 2020 because nobody could fix the COBOL. The infrastructure is fragile and the experts are retiring.

How Fast Is Each Layer of Financial Infrastructure – and Why Does It Matter?

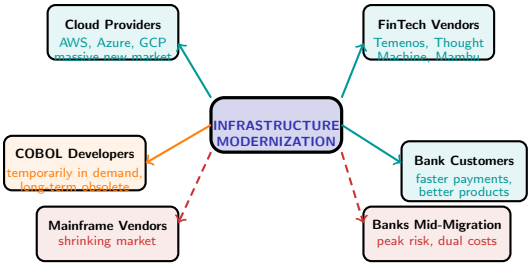


Reading the lollipop chart

- The horizontal axis uses a **log scale**. Each tick represents a 10x increase in processing time. The gap between Card Networks (2 seconds) and CSD Settlement (1 day) spans five orders of magnitude.
- **Dot color** encodes daily transaction volume. Teal dots (high volume) are the fastest layers – card networks and instant payments process billions of transactions per day. Purple dots (low volume) are the slowest: CSD settlement handles fewer but larger transfers.
- **Core Banking Batch** at 12 hours is the bottleneck. It sits between fast payment initiation and slow final settlement. This is the nightly batch window where your bank updates the actual ledger.
- **RTGS at 2 hours** is faster than core banking batch because central banks run real-time gross settlement. The irony: the central bank settles faster than your retail bank credits your account.
- **The modernization gap:** Card networks and instant payments already operate in seconds. Core banking and CSD settlement remain in the hours-to-days range. Closing this gap is the central challenge.

Illustrative estimates based on BIS (Bank for International Settlements, Basel 1930), SWIFT, and Visa (Foster City CA, 1958) benchmarks. SEPA Instant launched Nov 2017; FedNow launched Jul 2023. Processing times span five orders of magnitude – from 2 seconds (card networks) to 1 day (CSD settlement).

Who Wins and Who Loses as Legacy Infrastructure Gets Replaced?



Winners:

- **Cloud providers** gain a massive new market as banks migrate from on-premises mainframes to cloud infrastructure
- **FinTech vendors** (Temenos = Geneva 1993; Thought Machine = London 2014, founded by ex-Google Paul Taylor; Mambu = Berlin 2011) sell modern core banking platforms as replacements
- **Customers** eventually get faster payments, better products, and real-time account updates

Losers:

- **Mainframe vendors** see their installed base shrink as banks move to cloud-native architectures
- **Banks mid-migration** bear enormous risk: they run two systems simultaneously, doubling costs while exposing themselves to TSB-style failures

Mixed:

- **COBOL developers** are in peak demand now (rates of $100+/hour) as banks maintain legacy systems – but their skills become obsolete once migration completes

Infrastructure modernization creates clear winners (cloud, FinTech, customers) and losers (mainframe vendors, banks mid-migration). COBOL developers are the most ironic stakeholder: in demand precisely because the world is trying to replace them.

Four Questions That Reveal Whether Your Bank's Infrastructure Is a Strength or a Liability

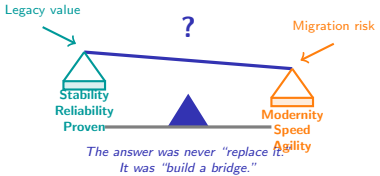
When evaluating any bank's technology infrastructure – whether as a regulator, investor, or board member – ask these four questions:

1. How old is the core system, and when was it last updated?
If the core banking platform was installed before 2000 and has not been re-platformed, it is running on architecture designed for batch processing. Every year of delay increases migration cost and risk.

2. How many people understand it, and what is their average age?
If fewer than five people understand the core system and their average age exceeds 55, the bank faces knowledge extinction risk. When these engineers retire, the system becomes a black box.

3. Is the migration plan big-bang or gradual?
Big-bang migrations (TSB) carry catastrophic risk. Gradual "strangler fig" approaches wrap new systems around old ones, migrating one product at a time. Slower but safer.

4. What is the 24-hour downtime recovery plan?
If the core system goes down for 24 hours, what happens? Can the bank operate manually? Do customers lose access? Is there a tested disaster recovery procedure, or just a plan on paper?



The framework in practice:
These four questions map directly to the key decisions every bank board faces:

- **Q1 + Q2** reveal the urgency: how close is the system to becoming unmaintainable?
- **Q3** determines the risk profile: big-bang migration vs. gradual strangler fig
- **Q4** tests operational resilience: can the bank survive while the transition is underway?

These four questions work for any bank, any jurisdiction. They separate banks that manage their technical debt from those that are one retirement or one failed migration away from crisis.

Should a Mid-Sized Swiss Bank Migrate to Cloud or Keep the Mainframe?

Mini-Challenge (15 minutes)

A Swiss Kantonalbank (cantonal publicly-guaranteed bank, 24 active, e.g. ZKB Zurich) runs its core banking on an IBM mainframe installed in 1998. Three COBOL developers maintain it (average age: 62). A vendor proposes a 3-year cloud migration to a modern platform (Avaloq = Zurich 1985 / Temenos = Geneva 1993). The board must decide: migrate, maintain, or hybrid?

Apply the four evaluation questions:

1. **How old, when last updated?** Installed 1998, last major update 2011. Running on IBM z/OS with DB2. Batch-only settlement. The system predates mobile banking, instant payments, and open banking APIs. Every new feature requires a COBOL wrapper.
2. **How many understand it, what age?** Three COBOL developers, ages 59, 63, 64. No succession plan. The youngest will retire in 6 years. When they leave, the bank loses the ability to maintain its own core system. Hiring replacements is nearly impossible.
3. **Big-bang or gradual?** The vendor proposes a phased approach: migrate retail accounts first (18 months), then corporate (12 months), then treasury (6 months). But TSB tried phased and still failed. What safeguards does the Kantonalbank need?
4. **24-hour downtime plan?** Current disaster recovery: switch to a backup mainframe in Zurich (tested annually). Cloud migration adds a second failure mode. During the 3-year transition, the bank runs two systems – doubling the attack surface.

Discuss: Migrate, maintain, or hybrid? What would you recommend to the board – and what is the one risk you would insist they address first?

"The answer was never 'replace the mainframe.' It was 'build a bridge to the future while respecting the past.'"

Every bank faces this decision. The four-question framework separates banks that manage their technical debt strategically from those that are one retirement away from crisis.