L07: Operational Resilience & Automated Stress Testing

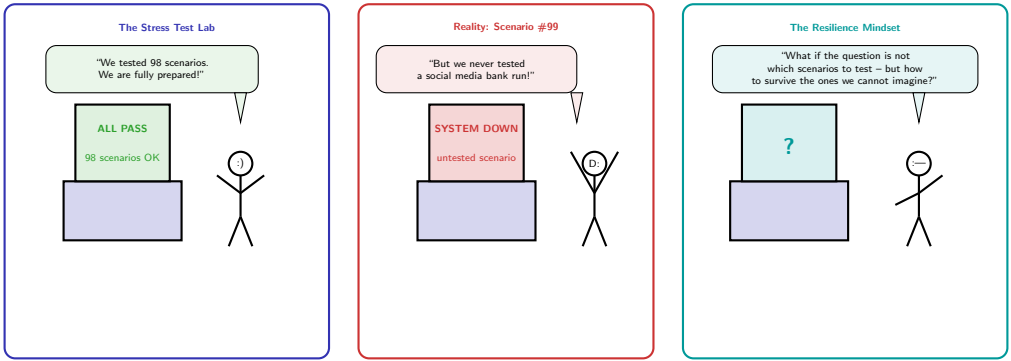
Extended Slides – BSc Digital Finance Course

Digital Finance

What Will You Be Able to Do After This Lecture?

1. Model operational loss distributions using the Compound Poisson process and estimate tail risk via Extreme Value Theory
2. Design forward and reverse stress test scenarios using correlated Monte Carlo simulation
3. Implement a loss simulation engine and reverse stress test solver in Python
4. Calculate operational risk capital under the Basel III Standardised Measurement Approach (SMA)
5. Evaluate automated stress test infrastructure including RPO/RTO dependency analysis
6. Assess institutional DORA compliance readiness using the resilience maturity framework

Six objectives: formal models (1–2), Python code (3, 5), regulatory capital (4), applied evaluation (6). Combines 12 charts with working code.



You stress-test for the last crisis. But the next one is always different.

How Do You Model Operational Losses When Both Frequency and Severity Are Random?

Compound Poisson loss model:

$$L = \sum_{i=1}^{N} X_i, \quad N \sim \text{Poisson}(\lambda), \quad X_i \sim F_X \text{ (severity)}$$

Moments of the aggregate loss:

$$\mathbb{E}[L] = \lambda \cdot \mathbb{E}[X], \quad \text{Var}(L) = \lambda \cdot \mathbb{E}[X^2]$$

Key insight: The tail of L is driven by severity F_X, not frequency λ.

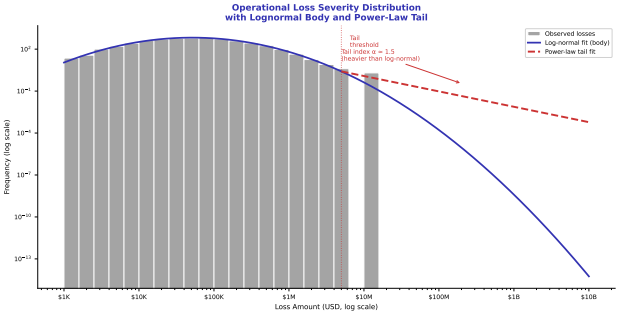
Common severity distributions:

- **Lognormal:** $\ln X \sim \mathcal{N}(\mu, \sigma^2)$ – good for body, underestimates tail
- **Generalised Pareto (GPD):** $G_{\xi,\beta}(x) = 1 - (1 + \xi x/\beta)^{-1/\xi}$ – captures heavy tail
- **Splice model:** Lognormal below threshold u, GPD above – best of both

Loss Distribution Approach (LDA): Convolve frequency and severity via Monte Carlo to obtain the full loss distribution. Capital $= \text{VaR}_{99.9\%}(L)$.

The compound Poisson model separates "how often" from "how bad." The capital requirement is dominated by severity, not frequency.

What Shape Does an Operational Loss Distribution Actually Take?



Simulated data based on industry loss parameters. Tail divergence illustrates excess kurtosis in operational risk.

- The log-scale histogram reveals the body-tail disconnect: most losses cluster below **$**1M, but the tail extends to billions
- The **lognormal fit** (purple) captures the body well but underestimates the right tail
- The **power-law tail** (red dashed) diverges upward above the **$**5M threshold – operational losses are heavier-tailed than lognormal
- The tail index $\alpha \approx 1.5$ means variance may be infinite – standard risk measures break down
- This justifies the splice approach: lognormal body + GPD tail

Operational losses look manageable in the body but catastrophic in the tail. Power-law divergence above $5M is why EVT matters.

Can You Simulate a Full Operational Loss Distribution in 18 Lines?

```python
import numpy as np
from scipy.stats import poisson, lognorm

def compound_poisson_lda(
    freq_lambda, sev_mu, sev_sigma,
    n_sims=100_000):
    """Compound Poisson LDA simulation."""
    total = np.zeros(n_sims)
    for i in range(n_sims):
        n_events = poisson.rvs(freq_lambda)
        if n_events > 0:
            sev = lognorm.rvs(
                s=sev_sigma,
                scale=np.exp(sev_mu),
                size=n_events)
            total[i] = sev.sum()
    return total

losses = compound_poisson_lda(5, 14.5, 1.5)
```

How it works

- **Line 9:** Draw event count from Poisson(λ) – "how many losses this year?"
- **Lines 11–14:** Draw each severity from lognormal – "how bad is each loss?"
- **Line 15:** Sum severities to get aggregate annual loss
- **Repeat** 100,000 times to build the full distribution
- From the distribution: $\text{VaR}_{99.9\%}$ gives capital, ES gives tail average

Extension: Replace lognormal with GPD above a threshold to capture the heavy tail more accurately.

18 lines of Python produce a full operational loss distribution. Parameter σ controls the tail – increase it and watch capital requirements explode.

Which Categories of Operational Loss Actually Cost the Most?

Operational Loss Event Types: Basel Hierarchy
(Inner = Category, Outer = Sub-type, Size = Avg Loss Share)

Illustrative shares based on ORX/Basel operational risk data. Proportions are approximate.

- **Inner ring:** Basel event categories; size = share of total losses
- **Outer ring:** Sub-types within each category
- **Clients & Products** dominates (38%) – mis-selling, market manipulation
- **Execution & Delivery** (22%) – transaction errors, model failures
- **External Fraud** (14%) – cyber attacks alone are 60% of this category

Conduct risk dwarfs all other categories. Transaction errors and vendor failures collectively rival external fraud.

Why Do Standard Distributions Fail in the Tail – and What Replaces Them?

Pickands–Balkema–de Haan theorem:
For losses X above a high threshold u, the excess distribution converges to the GPD:

$$F_u(x) = P(X - u \le x \mid X > u) \xrightarrow{u \to \infty} G_{\xi,\beta}(x) = 1 - \left(1 + \frac{\xi x}{\beta}\right)^{-1/\xi}$$

Shape parameter ξ controls the tail:

- $\xi > 0$: Heavy tail (Pareto-like) – operational risk, catastrophes
- $\xi = 0$: Exponential tail – moderate risks
- $\xi < 0$: Bounded tail – physical constraints

VaR and ES from GPD:

$$\mathrm{VaR}_\alpha = u + \frac{\beta}{\xi}\left[\left(\frac{n}{n_u}(1-\alpha)\right)^{-\xi} - 1\right]$$

$$\mathrm{ES}_\alpha = \frac{\mathrm{VaR}_\alpha + \beta - \xi u}{1 - \xi}, \quad \xi < 1$$

EVT does not assume a distribution – it lets the data tell you the tail shape. For operational risk, $\xi \approx 0.5$–1.5, meaning tails are heavier than lognormal.

How Do Basel's Seven Risk Categories Map to Modern Digital Threats?

Basel Category	Traditional Example	Digital-Era Threat
Internal fraud	Rogue trader	Insider data exfiltration
External fraud	Card skimming	Deepfake CEO fraud
Employment	Discrimination suit	Remote work liability
Clients & products	Mis-selling	Algorithmic bias in lending
Physical assets	Fire, flood	Data centre climate risk
Business disruption	Power outage	Cloud provider failure
Execution & delivery	Settlement error	Smart contract bug

The mapping problem: Basel categories were designed in the 1990s for physical banking. Modern digital threats (AI bias, cloud concentration, deepfake fraud) cut across multiple categories simultaneously.

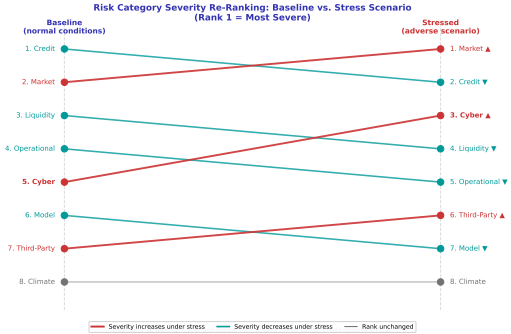
Why categorisation matters

- Capital is allocated *per category* under the SMA – misclassification means mispriced capital
- A cloud outage is "business disruption" but triggers "execution & delivery" failures and "client" losses simultaneously
- AI-related losses have no dedicated category – they scatter across fraud, clients, and execution
- DORA adds a cross-cutting ICT risk layer that overlays all seven categories

The honest answer: The taxonomy is a regulatory convenience, not a natural boundary. Real operational failures cascade across categories.

Basel's seven categories are a useful map but incomplete. Digital threats cross boundaries – one incident can trigger losses across multiple categories.

How Does Stress Re-Rank the Risks You Thought You Understood?



- **Red lines:** risks that *increase* in severity under stress – cyber jumps from rank 5 to rank 3
- **Teal lines:** risks that *decrease* relatively – liquidity drops as credit dominates
- **The key insight:** stress does not uniformly amplify all risks; it reshuffles the ranking
- Under normal conditions, credit dominates. Under stress, cyber and market risks surge
- This re-ranking is why static risk appetites fail – the hierarchy changes with the environment

The slope chart exposes a dangerous assumption: risk rankings are not stable. Under stress, cyber jumps two ranks – the hierarchy you planned for does not hold.

Can You Generate Internally Consistent Stress Scenarios in 18 Lines?

```python
import numpy as np

def correlated_scenarios(
    corr_matrix, n_scenarios=10_000,
    seed=42):
    """Cholesky-based correlated scenario
    generator for stress testing."""
    np.random.seed(seed)
    n_factors = corr_matrix.shape[0]
    L = np.linalg.cholesky(corr_matrix)
    Z = np.random.normal(
        size=(n_scenarios, n_factors))
    return Z @ L.T # correlated normals

# 4 risk factors: credit, market, op, cyber
rho = np.array([
    [1.0, 0.6, 0.3, 0.2],
    [0.6, 1.0, 0.4, 0.3],
    [0.3, 0.4, 1.0, 0.7],
    [0.2, 0.3, 0.7, 1.0]])
scenarios = correlated_scenarios(rho)
```

How it works

- **Cholesky decomposition:** $\Sigma = LL^\top$ transforms independent normals into correlated ones
- **Line 10:** Decompose the correlation matrix into lower triangular L
- **Line 12:** Multiply independent draws by $L^\top$ to induce correlation
- The 0.7 correlation between operational and cyber risk means a cyber shock almost always co-occurs with operational disruption
- Output: 10,000 internally consistent scenarios ready for simulation

Extension: Use t-copula instead of Gaussian for heavier tail dependence.

Cholesky ensures scenarios are internally consistent: cyber co-occurs with operational disruption ($\rho = 0.7$). Independent scenarios miss these dependencies.

Why Does the Choice of Copula Determine Whether Your Stress Test Is Realistic?

Sklar's theorem applied to operational risk:

$$F(l_1, \ldots, l_K) = C\big(F_1(l_1), \ldots, F_K(l_K)\big)$$

where l_k is the loss in Basel category k and C is the copula.

Gaussian copula: $\lambda_U = 0$ (zero upper tail dependence)

- Implies: extreme losses in one category are essentially independent of extreme losses in another
- This is *wrong* for operational risk – cyber attacks trigger execution failures which trigger client losses

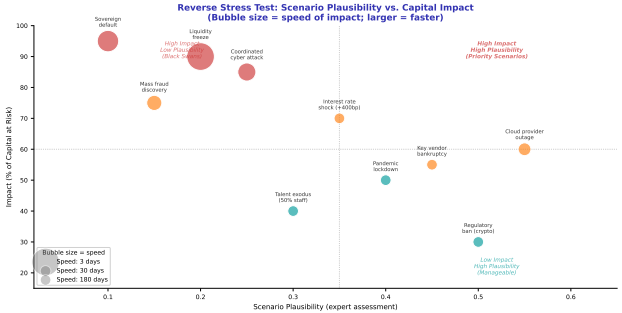
Student-t copula: $\lambda_U > 0$ (positive tail dependence)

$$\lambda_U = 2\, t_{\nu+1}\left(-\sqrt{\frac{(\nu+1)(1-\rho)}{1+\rho}}\right) > 0$$

Practical implication: Using a Gaussian copula underestimates the probability that multiple operational risk categories blow up simultaneously. The t-copula with low ν (4–6) captures the clustering seen in crises.

The copula choice determines whether risk categories blow up simultaneously. Gaussian says no; t-copula says yes. Crises prove the t-copula right.

Which Scenarios Are Both Plausible and Catastrophic?



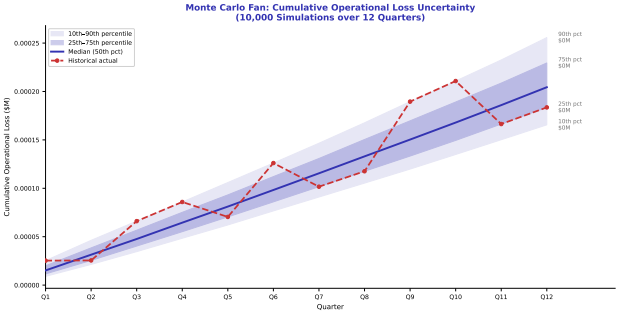
Illustrative reverse stress test output. Plausibility scores from expert panels; impact estimated via Monte Carlo simulation.

- **Upper-right quadrant** (high plausibility, high impact): priority scenarios that demand immediate attention
- **Bubble size:** speed of impact – larger bubbles hit faster, leaving less time to respond
- **Liquidity freeze** and **sovereign default**: highest impact but lower plausibility
- **Cloud outage** and **vendor bankruptcy**: moderate impact but highly plausible
- The **priority zone** is not the highest-impact corner – it is where plausibility meets severity

Key takeaway: Reverse stress tests prioritise by plausibility, not just severity. A plausible 60% capital hit matters more than an implausible 95% one.

The most dangerous scenarios are not the most extreme – they are both severe and plausible. Cloud outages sit in that sweet spot.

How Wide Is the Uncertainty in Cumulative Operational Losses?



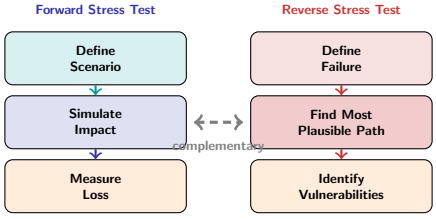
Simulated loss paths using lognormal increments with time-varying volatility. Widening bands reflect compounding uncertainty over the planning horizon.

- The fan widens over time: uncertainty compounds quarter by quarter
- The **median** (purple line) is the expected path – but no single realisation follows the median
- The **10th–90th percentile band** shows that outcomes at Q12 range from moderate to catastrophic
- The **historical actual** (red dashed) tracks above the median – demonstrating that tail events are not theoretical
- The widening gap between P25 and P75 shows that even "reasonable" outcomes diverge substantially

For capital planning: Use P90 or P99 for capital buffers, not the median.

The fan chart is honesty in visual form: it shows the range of outcomes, not just the expected one. Planning at the median is gambling; at P90 is prudence.

What Is the Difference Between Asking “How Bad?” and “What Kills Us?”



Two complementary approaches

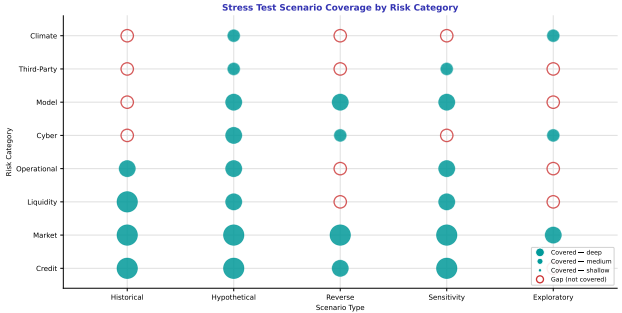
- **Forward:** Start with a scenario (“interest rates rise 400bp”), simulate the impact, measure the loss. Answers: “how bad could this get?”
- **Reverse:** Start with failure (“the bank is insolvent”), work backward to find the most plausible path. Answers: “what would kill us?”
- Forward tests validate known defences; reverse tests discover unknown vulnerabilities
- DORA and PRA require both: forward for known risks, reverse for existential threats

Why reverse matters more:

- Forward tests can only check scenarios you imagine
- Reverse tests reveal the shortest path to failure – which may be a scenario nobody considered

Forward tests check your defences against known threats. Reverse tests reveal the threats you did not know about. You need both.

Where Are the Blind Spots in Your Stress Test Programme?



Illustrative coverage assessment. Circle size indicates depth of scenario analysis.

- **Coverage assessment:** Each cell shows whether a risk category is tested under a given scenario type
- Credit and market risk: full coverage across all five scenario types
- **Climate risk:** Only hypothetical and exploratory scenarios exist – no historical data to calibrate
- **Third-party risk:** Covered only in hypothetical and sensitivity analysis – no reverse stress tests
- The diagonal pattern reveals that newer risks (cyber, climate) lack the historical scenarios that traditional risks benefit from

Implication: Automated scenario generators should target the gaps – generate synthetic historical scenarios for categories that lack real data.

The dot matrix is a diagnostic: it shows what you are not testing as clearly as what you are. The gaps are where the next crisis comes from.

How Do You Convolve Frequency and Severity Into a Single Capital Number?

Loss Distribution Approach (LDA):
For each Basel event type k, the annual aggregate loss is:

$$L_k = \sum_{i=1}^{N_k} X_{k,i}, \quad N_k \sim \text{Poisson}(\lambda_k), \quad X_{k,i} \sim F_{X_k}$$

Total operational loss: $L = \sum_{k=1}^{7} L_k$ (assuming independence or copula dependence)

Capital requirement:

$$K_{\text{OpRisk}} = \text{VaR}_{99.9\%}(L) - \mathbb{E}[L]$$

Analytic convolution is intractable for heavy-tailed severities. Monte Carlo:

1. For each simulation $s = 1, \ldots, S$: draw $N_k^{(s)}$, then draw $X_{k,1}^{(s)}, \ldots, X_{k,N_k^{(s)}}^{(s)}$
2. Compute $L^{(s)} = \sum_k \sum_i X_{k,i}^{(s)}$
3. $\widehat{\text{VaR}}_{99.9\%}$ = empirical 99.9th percentile of $\{L^{(s)}\}_{s=1}^{S}$

The LDA convolves seven frequency-severity pairs into one capital number. Analytic solutions fail for heavy tails – Monte Carlo is the only practical approach.

Can You Find the Scenario That Kills the Bank Using Binary Search?

```python
import numpy as np

def reverse_stress_search(
    loss_fn, capital, tol=0.01,
    lo=0.0, hi=10.0, max_iter=50):
    """Binary search for the smallest
    shock that breaches capital."""
    for _ in range(max_iter):
        mid = (lo + hi) / 2
        loss = loss_fn(mid)
        if abs(loss - capital) < tol:
            return mid
        if loss < capital:
            lo = mid
        else:
            hi = mid
    return mid

# Example: loss_fn maps shock to loss
def simple_loss(shock):
    return 50 * np.exp(0.8 * shock)

threshold = reverse_stress_search(
    simple_loss, capital=500)
```

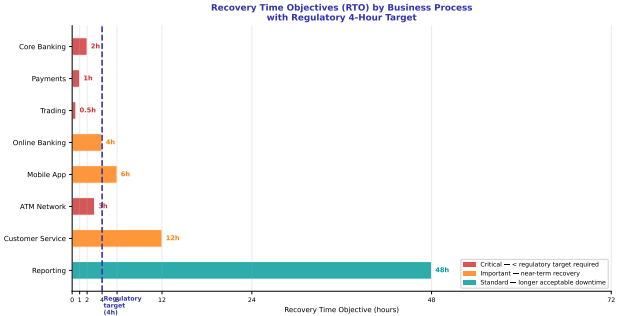
How it works

- **Goal:** Find the *smallest* shock magnitude that causes loss $\geq$ capital
- **Binary search:** Narrow the interval [lo, hi] until loss $\approx$ capital threshold
- **loss_fn:** Any function mapping shock severity to portfolio loss – can wrap a full Monte Carlo engine
- Converges in $\sim$30 iterations ($\log_2(10/0.01)$)
- The output is the "reverse VaR" – the minimum scenario that destroys the bank

Extension: Replace scalar shock with multivariate (use optimisation over correlated factors).

Reverse stress testing is an optimisation problem: find the smallest shock that breaches capital. Binary search is simple; gradient methods scale to high dimensions.

How Fast Must Each Business Process Recover – and Which Ones Fail the Target?



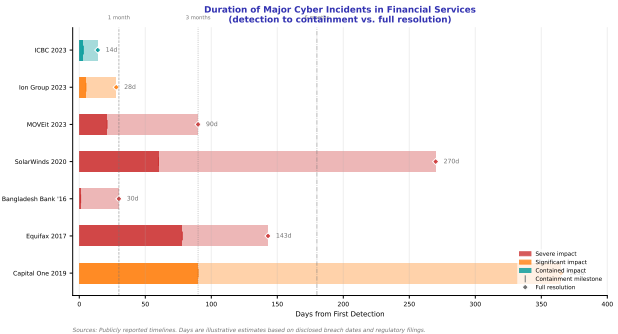
RTOs are illustrative based on typical financial institution BCP frameworks. Actual targets depend on regulatory jurisdiction and institution size.

- **Red bars:** Critical processes that must recover fastest – trading (0.5h), payments (1h), core banking (2h)
- **Purple dashed line:** Regulatory 4-hour target for critical services
- **Orange bars:** Important but non-critical – online banking (4h), mobile app (6h)
- **Reporting** (48h) has the longest acceptable downtime – it is not customer-facing
- The gap between RTO and *actual* recovery time is the resilience risk – tested recovery often exceeds the target

DORA impact: Regulators now require proof that RTOs are achievable, not just documented.

RTOs are promises. The Gantt reveals which the institution can keep (trading: 30 min) and which it probably cannot (reporting: 48h on paper, longer in practice).

How Long Does It Actually Take to Detect, Contain, and Resolve a Cyber Incident?



Sources: Publicly reported timelines. Days are illustrative estimates based on disclosed breach dates and regulatory filings.

- **Dark bars:** Time from detection to containment – the "active attack" phase
- **Light bars:** Full resolution timeline including forensics, remediation, and regulatory reporting
- **SolarWinds** (2020): 60 days to contain, 270 days to resolve – supply chain attacks are the slowest
- **ICBC** (2023): Contained in 3 days – faster response reflects improved playbooks
- The 1-month and 3-month reference lines show that most severe incidents exceed both

DORA requires incident reporting "without undue delay" – typically interpreted as 4–24 hours.

The timeline reveals the gap between containment and resolution: you stop the bleeding quickly, but full recovery takes months. DORA reporting falls in the first hours.

How Does Basel III Calculate Operational Risk Capital Without Internal Models?

Standardised Measurement Approach (SMA):

$$\text{BIC} = \max\Big(\text{II\&L Component},\ \text{SC Component},\ \text{FC Component}\Big)$$

$$\text{BIC} = \begin{cases} 0.12 \times \text{BI} & \text{if BI} \leq \text{€1bn} \\ 0.15 \times \text{BI} & \text{if BI} \in (\text{€1bn}, \text{€30bn}] \\ 0.18 \times \text{BI} & \text{if BI} > \text{€30bn} \end{cases}$$

Internal Loss Multiplier (ILM):

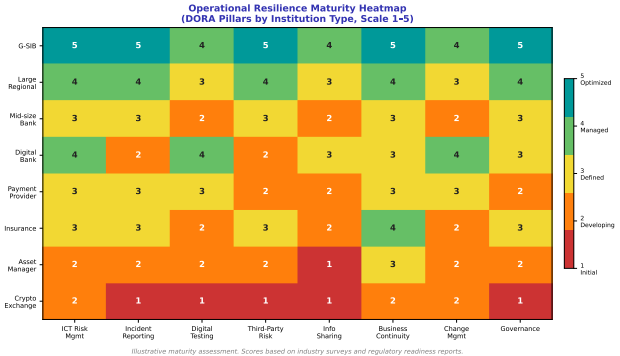
$$\text{ILM} = \ln\left(e^1 - 1 + \left(\frac{\text{LC}}{\text{BIC}}\right)^{0.8}\right)$$

OpRisk Capital $= \text{BIC} \times \text{ILM}$

Key feature: SMA replaces all previous approaches (BIA, TSA, AMA) with a single formula. Banks with high historical losses (LC) face a multiplier > 1; banks with low losses get a discount.

SMA is deliberately simple: BIC (size-based) multiplied by ILM (loss-history). No internal models – but also no reward for sophisticated risk management.

How Ready Is Each Institution Type for Operational Resilience Regulation?



- **G-SIBs** (top row): near-optimal across all pillars – they have the resources and regulatory pressure
- **Crypto exchanges** (bottom row): lowest maturity across the board – minimal governance, testing, and information sharing
- **Digital banks** show an interesting pattern: strong in testing and change management, weak in third-party risk and incident reporting
- The **diagonal gap**: institutions that score well on technology (testing, change) often score poorly on governance (reporting, oversight)

Implication: Maturity is not uniform – institutions have peaks and valleys that reveal their strategic priorities.

Resilience maturity is a profile, not a single score. Digital banks lead in testing but lag in governance; G-SIBs show the reverse pattern.

Can You Calculate Recovery Objectives from a Dependency Graph?

```python
import numpy as np

def calc_rto(deps, rto_base):
    """Calculate effective RTO from
    dependency graph (topological)."""
    n = len(rto_base)
    eff_rto = rto_base.copy()
    # Propagate: RTO >= max(dep RTOs)
    changed = True
    while changed:
        changed = False
        for i in range(n):
            for j in deps.get(i, []):
                new = max(eff_rto[i],
                          eff_rto[j])
                if new > eff_rto[i]:
                    eff_rto[i] = new
                    changed = True
    return eff_rto

# Services: 0=payments, 1=core, 2=cloud
deps = {0: [1], 1: [2]} # pay->core->cloud
base = np.array([1.0, 2.0, 4.0]) # hours
effective = calc_rto(deps, base)
```

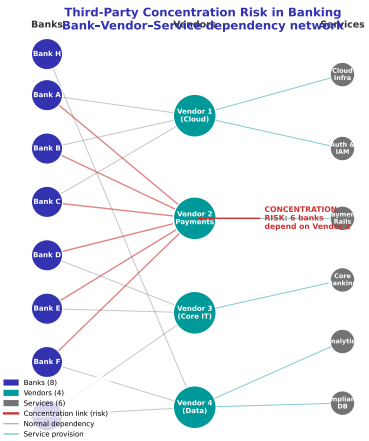
How it works

- **Dependency graph:** Each service depends on others – payments depend on core banking, which depends on cloud
- **Propagation:** A service cannot recover faster than its slowest dependency
- **Result:** Payments has a base RTO of 1 hour, but its effective RTO is 4 hours (limited by cloud)
- This reveals hidden bottlenecks: the service you think recovers in 1 hour actually takes 4

Extension: Add RPO (Recovery Point Objective) – how much data can you lose? Propagate similarly through the dependency graph.

Dependency propagation reveals hidden bottlenecks: a 1-hour RTO that depends on a 4-hour service actually has a 4-hour effective RTO. The dependency graph never lies.

Where Does Concentration Risk Hide in Your Vendor Network?



Illustrative network. Node size proportional to role scope. Red edges indicate single-point-of-failure dependency.

- **Red arrows:** 6 of 8 banks depend on Vendor 2 (Payments) – concentration risk
- If Vendor 2 fails, 75% of banks lose payment capability simultaneously
- **DORA** requires monitoring of "critical ICT third-party providers" (CTPPs)
- Network structure reveals risks invisible in bilateral vendor assessments

Key question: If your top vendor failed tomorrow, how many critical services survive?

The network reveals what bilateral assessments miss: systemic concentration. When 6 banks share one vendor, a single failure becomes everyone's crisis.

How Do You Measure the Average Loss in the Worst Scenarios?

Conditional Expected Shortfall under scenario S:

$$\mathrm{ES}_\alpha^S = \mathbb{E}\big[L \mid L \geq \mathrm{VaR}_\alpha^S\big] = \frac{1}{1-\alpha}\int_\alpha^1 \mathrm{VaR}_u^S \, du$$

Scenario-conditional loss:

$$L^S = \sum_{k=1}^{K} w_k \cdot L_k \cdot \mathrm{stress}_k^S$$

where stress_k^S is the scenario-specific multiplier for risk category k.

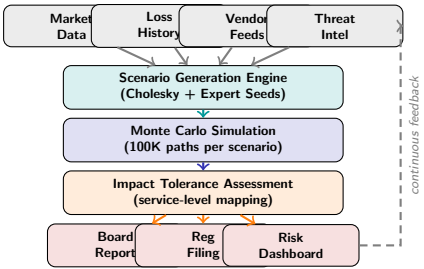
Practical computation:

1. Simulate L^S under each scenario S using correlated Monte Carlo
2. Compute ES as the mean of losses exceeding VaR: $\widehat{\mathrm{ES}}_\alpha^S = \frac{1}{|\{s : L^{(s)} \geq \widehat{\mathrm{VaR}}_\alpha\}|} \sum_{s : L^{(s)} \geq \widehat{\mathrm{VaR}}_\alpha} L^{(s)}$
3. Compare across scenarios to rank by tail severity

Key property: ES is coherent (subadditive) – diversification always reduces risk.

ES answers the question VaR cannot: given we are in the tail, how bad is it on average? Scenario-conditional ES ranks scenarios by tail severity, not probability.

What Does an End-to-End Automated Stress Test Architecture Look Like?



Five-layer architecture

- **Layer 1 – Data:** Market feeds, internal loss history, vendor status, and threat intelligence flow in continuously
- **Layer 2 – Scenarios:** Cholesky-correlated generation produces thousands of internally consistent scenarios
- **Layer 3 – Simulation:** Each scenario runs through a Monte Carlo engine with 100K+ paths
- **Layer 4 – Impact:** Losses are mapped to business services and compared against impact tolerances
- **Layer 5 – Output:** Board reports, regulatory filings, and real-time dashboards

The feedback loop: Dashboard findings feed back into threat intelligence, closing the cycle.

Automated stress testing is a pipeline: data in, scenarios generated, simulations run, impacts assessed, and results feed back into the next cycle.

How Does a Single Vendor Failure Cascade Through the Financial Network?

Cascade failure model:
Let $x_i(t) \in \{0, 1\}$ be the operational status of institution i at time t. Failure propagates via:

$$x_i(t+1) = \begin{cases} 0 & \text{if } \sum_{j \in \mathcal{N}(i)} w_{ij} \cdot (1 - x_j(t)) \geq \theta_i \\ x_i(t) & \text{otherwise} \end{cases}$$

where w_{ij} is the dependency weight, $\mathcal{N}(i)$ is the neighbour set, and θ_i is the failure threshold.

Systemic risk measure:

$$\text{SRI} = \frac{1}{N} \sum_{i=1}^{N} P\left[\sum_{j \in \mathcal{N}(i)} w_{ij}(1 - x_j) \geq \theta_i \right]$$

Key insight: A highly connected vendor with low θ (fragile) can trigger cascading failures even if its individual failure probability is low. Network topology matters more than individual reliability.

The cascade model shows systemic risk depends on network structure, not individual robustness. A single vendor failure propagates until the cascade dies or the system collapses.

Can You Fit a GPD to Operational Loss Tails in 18 Lines?

```python
import numpy as np
from scipy.stats import genpareto
from scipy.optimize import minimize

def fit_gpd_tail(losses, threshold):
    """Fit GPD to excesses over threshold
    using maximum likelihood."""
    excesses = losses[losses > threshold]
    excesses = excesses - threshold
    # MLE fit via scipy
    xi, loc, beta = genpareto.fit(
        excesses, floc=0)
    n = len(losses)
    n_u = len(excesses)
    return xi, beta, n, n_u

def gpd_var(alpha, xi, beta, n, n_u, u):
    """VaR from GPD tail estimate."""
    return u + (beta/xi) * (
        (n/n_u * (1-alpha))**(-xi) - 1)

losses = np.random.lognormal(14, 1.5, 5000)
xi, beta, n, nu = fit_gpd_tail(losses, 5e6)
var99 = gpd_var(0.999, xi, beta, n, nu, 5e6)
```

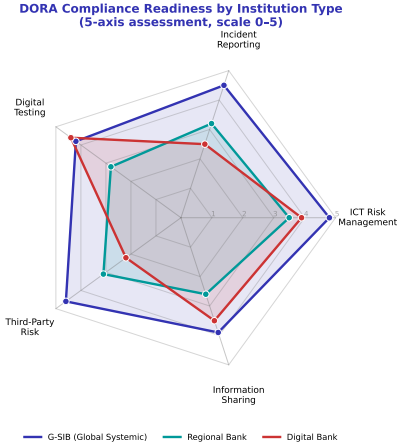
How it works

- **Threshold selection:** Choose u where the mean excess plot becomes linear (typically the 90th–95th percentile of losses)
- **Excesses:** Extract observations above u and subtract u
- **MLE fit:** Estimate shape (ξ) and scale (β) via maximum likelihood
- **VaR formula:** Plug GPD parameters into the closed-form VaR expression
- A positive ξ confirms heavy tails – standard distributions underestimate risk

Practical tip: Run stability analysis – vary u and check that ξ remains approximately constant.

EVT lets the data speak: fit the GPD to excesses above a threshold and extract the tail shape. A positive ξ means standard VaR is wrong.

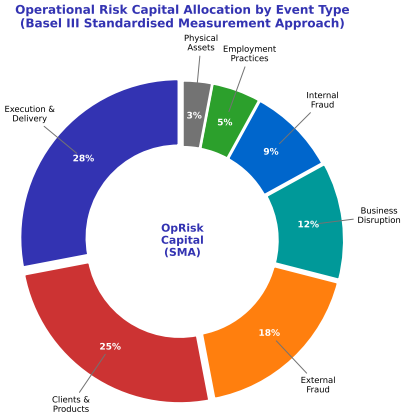
How Do Different Institution Types Compare on DORA Readiness?



- **G-SIBs** (purple): Near-5.0 on ICT risk and third-party risk; weaker on information sharing (3.9)
- **Regional banks** (teal): Mid-range 3.0–3.5 across all axes
- **Digital banks** (red): Strong on testing (4.4) but weakest on third-party risk (2.2)
- **Information sharing** is lowest for all types – cultural resistance to disclosing vulnerabilities

No institution type is uniformly DORA-ready. G-SIBs lead on governance but lag on sharing; digital banks lead on testing but lag on third-party risk.

Where Does Operational Risk Capital Actually Go?



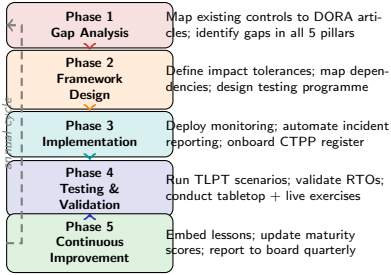
Illustrative allocation based on industry loss data. SMA replaced all previous approaches under Basel III finalisation.

- **Execution & Delivery** (28%): Transaction errors, model failures, vendor issues
- **Clients & Products** (25%): Mis-selling and suitability failures
- **External Fraud** (18%): Cyber attacks and card fraud
- **Business Disruption** (12%): IT outages, cloud failures
- Smallest three categories total only 17% combined

SMA implication: Capital is driven by historical losses – past conduct fines create a permanent surcharge.

Over half of operational risk capital goes to execution errors and conduct risk – the daily failures of complex institutions, not exotic tail events.

What Does a DORA Compliance Roadmap Look Like for a Mid-Size Bank?

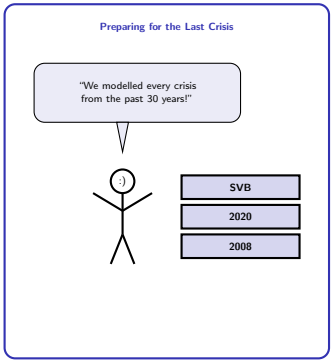


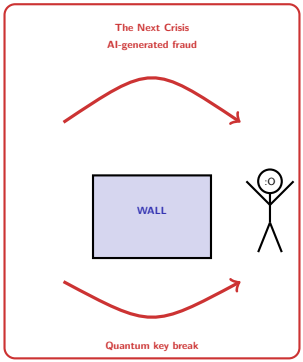
Five-phase roadmap

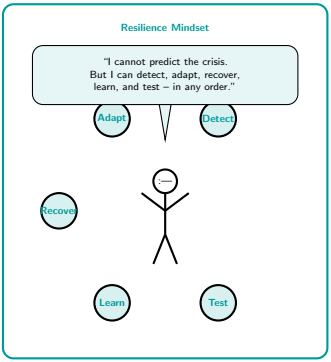
- **Phase 1 (3 months):** Gap analysis against all DORA articles – most mid-size banks find 40–60% coverage
- **Phase 2 (4 months):** Design the resilience framework – impact tolerances, dependency maps, governance model
- **Phase 3 (6 months):** Technical implementation – monitoring tools, automated reporting, CTPP register
- **Phase 4 (3 months):** Threat-Led Penetration Testing (TLPT) and live recovery exercises
- **Phase 5 (ongoing):** Continuous improvement cycle – lessons embedded, maturity tracked

Total timeline: 16–18 months from kickoff to steady state. The feedback loop makes it perpetual.

DORA compliance is a continuous cycle, not a project with an end date. Phase 5 feeds back into Phase 1, and the maturity score should improve each year.







You cannot build a wall tall enough. But you can build an organisation that survives what comes over it.