Challenges in Delivering Production-Grade Actuarial Applications

Building Custom Actuarial Applications on a Data Platform

June 2025

DL Lim

CONTENTS

1 EXECUTIVE SUMMARY

As insurers continue their digital transformation journeys, actuarial teams are being asked to **move beyond traditional spreadsheet-based tools** and vendor-dependent platforms toward scalable, **production-grade models built on modern data infrastructure**. This shift promises significant value: **reduced vendor lock-in, improved model governance, and better alignment with enterprise data strategies.**

When done well, it enables actuarial teams to own, control, and evolve their models — embedding them within the wider data ecosystem to enable **real-time analytics, reproducibility, and business agility.** Successful transformation journeys involve a blend of technical skills, supported by cultural, architectural, and organisational changes to ensure sustainable and full integration within the broader business strategy.

This whitepaper outlines the core challenges and pragmatic strategies for modernising actuarial applications. It examines key issues such as **fragmented talent pools, non-scalable legacy workflows, and the absence of integration and control in current environments.**

It proposes a **structured path forward**: building cross-functional delivery teams, refactoring legacy models for cloud-native scale, and embedding governance into the platform layer. It also highlights the need for education, upskilling, and stakeholder alignment to ensure adoption and sustained impact.

Ultimately, actuarial modernisation is not about replacing spreadsheets with Python — it is about transforming actuarial work into a transparent, auditable, and scalable part of the enterprise digital fabric.

2 INTRODUCTION

As digital transformation accelerates, actuarial teams are moving away from legacy Excel-based models to production-grade systems on modern data platforms. This shift enables **self-ownership of models**, reducing reliance on vendors and **mitigating lock-in risks**. By modernising workflows, organisations can centralise actuarial models, improve governance, and unlock **company-wide analytics**.

This paper explores the common problems in building and maintaining custom, production-grade actuarial applications and streamlining processes for posterity.

An actuarial application may refer to any of the following examples:

- Valuation models
- Reserving models
- Projection models
- Pricing tools
- Asset-liability management tools
- Stochastic and Monte-carlo simulators
- Solvency models
- Capital and risk models
- Experience studies

Typically and traditionally, these models and tools are built in Excel, traditional actuarial projection software from vendors, in-house tooling or other similar tooling.

This paper focuses on the conversion of these models off of Excel or vendor software, into a self-owned code model on a programming language of the organisation's choice — typically Python, R, SQL, Matlab etc. — with integration to the wider company's data platform.

3 PROBLEM STATEMENT

3.1 A FRAGMENTED TALENT POOL AND MISALIGNED MINDSETS

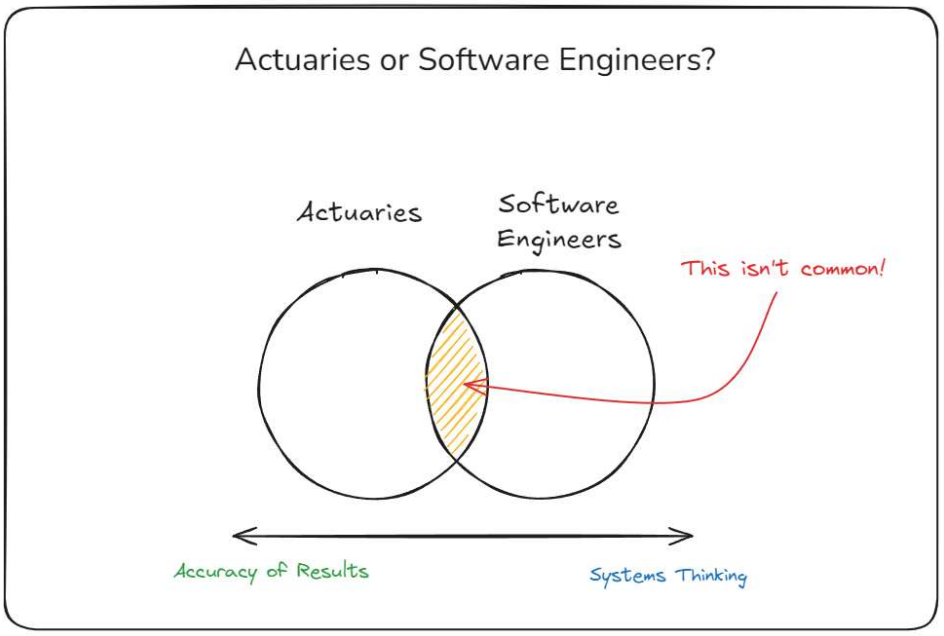
At the heart of actuarial application delivery lies a persistent capability divide: Do you find actuaries who can code in Python, or software engineers who understand reserving models, regulatory capital, or stochastic simulations? Often, you don't find either — you get one or the other.

But the issue runs deeper than recruitment — it's about **mindset and team composition**. In many insurance organisations undergoing transformation, actuarial and engineering teams **function in isolation**, leading to persistent disconnects in communication, alignment, and execution.

Production delivery requires cross-functional collaboration: a business SME, a data SME, cloud engineers and ideally, a hybrid specialist who speaks both actuarial and engineering fluently. But even in their absence, a structured pairing model with delivery discipline (scrum, documentation, code review, user acceptance testing) and a clear communication layer (project management) is essential.

This is not about learning Python syntax; it's about transforming an Excel-centric, one-off modeling mindset into reproducible, testable, and **scalable systems thinking**.

The ability to write code is a powerful and commendable skill for actuaries, empowering greater ownership of models and accelerating innovation. However, bridging the gap between coding and systems engineering is essential to evolve actuarial prototypes into scalable, production-grade applications.



3.2 LEGACY MODELING APPROACHES DON'T SCALE

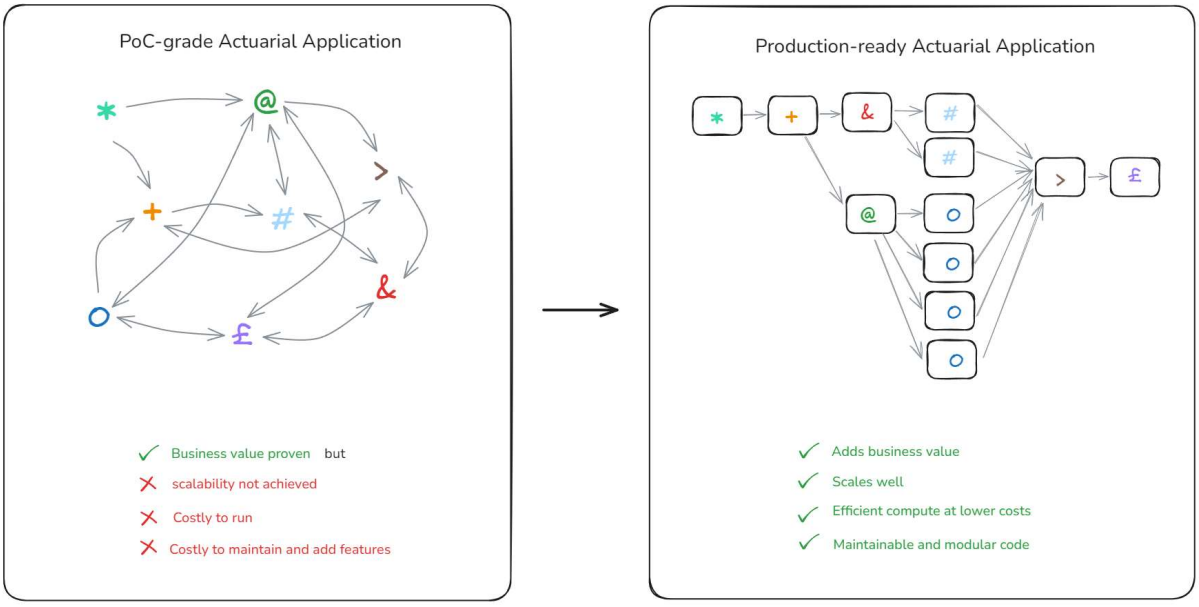
Actuarial models are often built for correctness, not performance. Legacy workflows rely heavily on nested loops, tabular lookups, and manual adjustments — not distributed computation, automated tests, or CI/CD pipelines. Transitioning to Databricks or any modern platform is not just a technical migration; it's a **paradigm shift**.

You move from "run this once for the regulator" to "run this nightly with lineage, audit, and monitoring." You can't do that with black-box models or one-off scripts. Model transparency, automated controls, and runtime efficiency (e.g., rewriting simulations for Spark, vectorising logic) are not optional — they're foundational.

This shift demands upskilling, but more importantly, **reframing actuarial logic into scalable, inspectable code.**

Realistically, upskilling takes a long time to gather the right experience and skills required for such an endeavour, and yet the business value has already been proven by the various proofs-of-concept.

There is also much operational risk leaving the entire IP in the hands of one or two individuals rather than having a production-ready team handle the day-to-day of the software.



3.3 INTEGRATION, CONTROL, AND TRUST IN A DISTRIBUTED WORLD

We often see actuarial teams bypassing a centralised system/software in favour of Excel spreadsheets shared via emails or drives for quicker and easier access. This is a symptom of needing speed of execution where many systems fail to provide the stability to do so.

Integrating actuarial logic into modern data platforms isn't just about code and data — it's about **governance**, which one would recognise is a part of the actuarial control cycle. Traditional actuarial work is rarely versioned or reviewed like production code. But production environments demand:

- Secure access to sensitive data (PII, policy history)
- Clear lineage from source to model to output
- Controlled deployment pipelines
- Audit trails and sign-offs

Without these, data becomes untraceable, and black-box models erode trust. Both actuaries and engineers fail to find a middle ground to maintain an application, which eventually falls into a state of disrepair or gets rebuilt from scratch, costing hundreds of thousands, if not millions.

The lack of direct integration into upstream and downstream systems is also a detriment to the effectiveness of the models — often formats need to be converted via spreadsheets from legacy actuarial software to modern reporting platforms — thus spawning a myriad unsystematic swamp of bespoke data conversion tools (which each have to be maintained)

Actuarial models don't live in isolation; they must be *embedded into data ecosystems* that can explain, reproduce, and scale them.

4 THE STRATEGIC AND REALISTIC PATH FORWARD

4.1 BUILD CROSS-FUNCTIONAL, EMBEDDED DELIVERY TEAMS

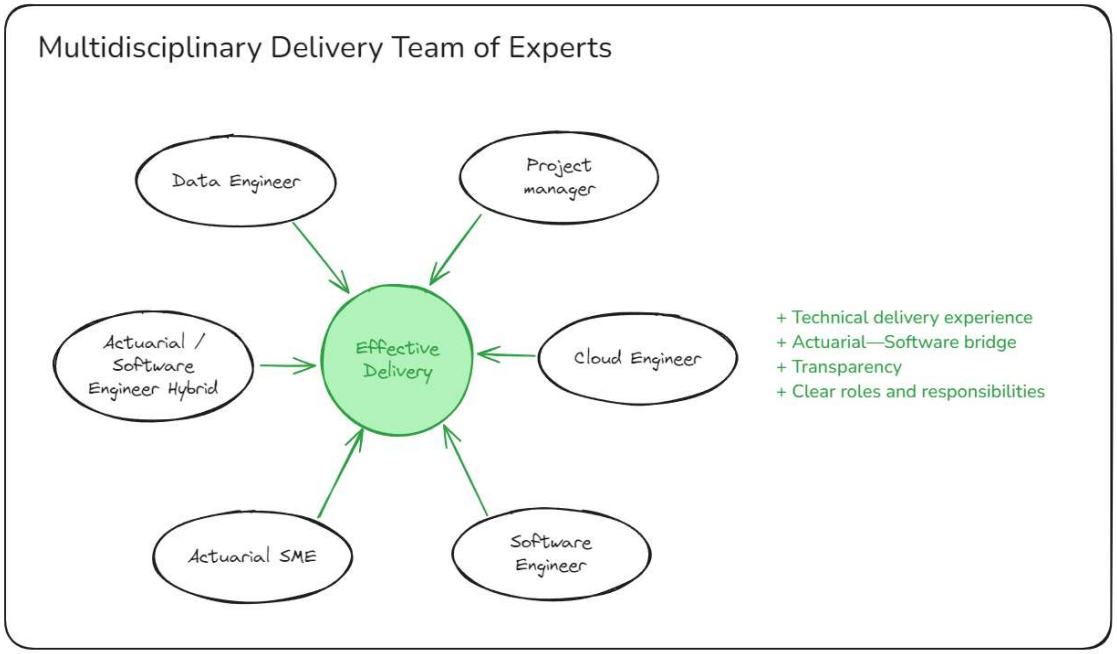
To bridge the capability and mindset divide, insurers must **embed actuaries within cross-functional product teams** — not siloed in their own functions. These teams should include:

- A business SME (e.g. actuarial or risk lead)
- A data engineer
- A cloud or DevOps engineer
- A delivery/project manager
- And ideally, a **hybrid actuarial engineer**

Where hybrid profiles are scarce, use structured **pairing and upskilling models** (e.g., pairing an actuary with an engineer in short-cycle sprints) and enforce **engineering discipline**: version control, code review, automated testing, and UAT.

This dynamic fosters a shared language between domains and builds sustainable, supportable systems.

Of course, in reality, roles and responsibilities may vary, team topologies, individual strengths and organisation dynamics set the pace of delivery. This proposed composition is a minimum framework for a high-performance team.

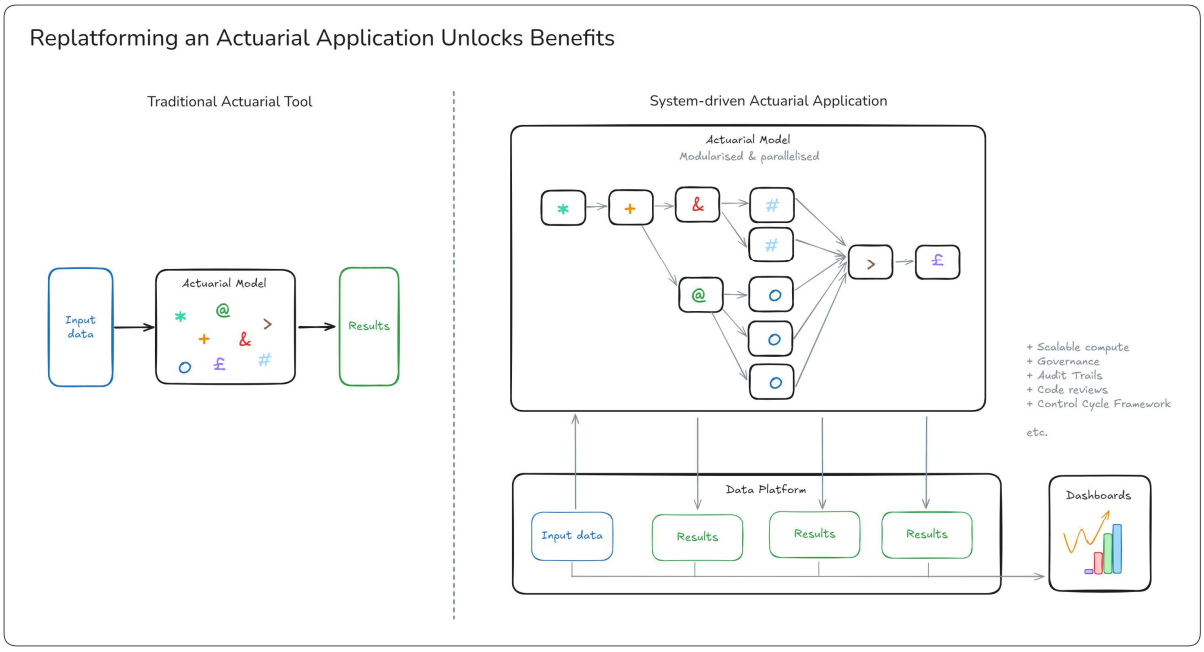


4.2 REPLATFORM AND REFACTOR LEGACY MODELS FOR SCALE

Rather than lift-and-shift legacy spreadsheets or scripts, organisations must **rethink the structure of their actuarial models** to align with modern compute paradigms:

- **Decompose large spreadsheets** into modular, testable functions
- Refactor simulations into **vectorised**, parallelisable logic
- Use **open-source libraries** (e.g., in Python or R) with strong test coverage
- Leverage **orchestration tools** (like Azure Data Factory, Databricks Workflows, Airflow, dbt)

Automate the entire model lifecycle with CI/CD pipelines, data validation, and monitoring hooks. This shifts actuarial work from point-in-time outputs to **continuous, observable processes** with full data lineage across the organisation.



4.3 OPERATIONALISE GOVERNANCE VIA PLATFORM-NATIVE CONTROLS

Trust in actuarial applications stems from **transparent, well-governed systems**. Embedding the control cycle into your platform architecture is critical. That means:

- Role-based access control (RBAC) and fine-grained data permissions
- Data lineage tracking (from source to model output)
- git-versioned code releases for reproduceability
- Audit logs for every model execution
- Approval and promotion workflows for production deployment

Use platform-native capabilities (e.g., Unity Catalog on Databricks, Azure Purview, or Snowflake's governance suite) instead of bolting governance on after the fact. Build once, **govern by design**, and ensure auditability is baked into every step — not added at the end.

4.4 BONUS: INVEST IN EDUCATION, UPSKILLING, AND STAKEHOLDER ALIGNMENT

No transformation succeeds without **people**. Actuaries, engineers, and business stakeholders all require targeted education to close capability gaps and align on expectations. This includes:

- **Upskilling actuaries** in software and data engineering fundamentals (e.g., Git, modular code, testing)
- **Training engineers** in actuarial concepts and paradigms - which often diverges from traditional tech delivery mindsets
- **Workshops** to foster shared understanding and improve general unity
- **Clear communication with senior leadership** to explain why these changes are essential for speed, auditability, and cost-effectiveness

Senior stakeholders must see this not as a tooling upgrade, but as a **strategic shift** in how actuarial value is delivered — one that requires ongoing investment in people and process, not just technology.

5 CHALLENGES AND CONSIDERATIONS

Despite having a clear path forward, it would be naive to overlook the complexity and intricacies of modernising insurance companies.

As we head down this path, here are some key challenges the industry faces as a whole:

5.1 CULTURAL RESISTANCE TO CHANGE AND LACK OF CLEAR STAKEHOLDER BUY-IN

One of the most persistent barriers to modernising actuarial systems is **not technology, but culture**. Many insurance organisations have long-standing ways of working — centred around Excel, manual workflows, and waterfall-style delivery. Shifting to a product-driven, platform-based, and code-centric model introduces a profound change to both day-to-day work and professional identity.

A key source of resistance is **fear — particularly around job security**. For actuaries and other domain experts who have built careers on specific tools and processes, the prospect of automation, code reviews, or cloud platforms can feel like a threat. The move from individual spreadsheet ownership to version-controlled, team-owned codebases may feel like a loss of autonomy or value. This creates friction, even among well-intentioned teams.

To address this, organisations must invest in **education and assurance**. Upskilling programs should be designed not just to teach tools like Python or Git, but to **frame change as an enabler**, not a threat — reinforcing that the goal is to amplify actuarial expertise, not replace it. Pairing programs, internal showcases, and shared success metrics can reinforce that message.

Resistance is often reinforced when **senior leadership lacks visibility** into the value of the transition. Without clear articulation of the benefits — such as faster iteration, better auditability, reduced risk, and greater cross-functional alignment — transformation efforts can be seen as "technical experiments" rather than strategic imperatives. Leaders need to understand that modernising actuarial delivery isn't just about efficiency — it's about **building competitive advantage through trust, speed, and scale**.

To succeed, change must be **led from the top, reinforced from the middle, and supported at the ground level**. That requires:

- A clear narrative that links platform investment to business outcomes
- Champions at every level of the organisation
- Feedback channels to ensure the transformation is inclusive and responsive

Culture change is slow, but without it, even the best tools and architectures will fail to take root.

5.2 CREATION OF NEW ROLES AND RESPONSIBILITIES

As actuarial systems evolve from siloed spreadsheets to integrated, production-grade applications, **the organisational chart must evolve with them**. Legacy team structures — where actuaries own the entire modelling lifecycle in isolation — are no longer sufficient to support scalable, governed, and maintainable solutions.

Modern actuarial delivery introduces **new responsibilities** that span technical, operational, and governance domains. Without clearly defined roles and ownership, these responsibilities often fall through the cracks, leading to brittle systems and unclear accountability.

Some critical new roles include:

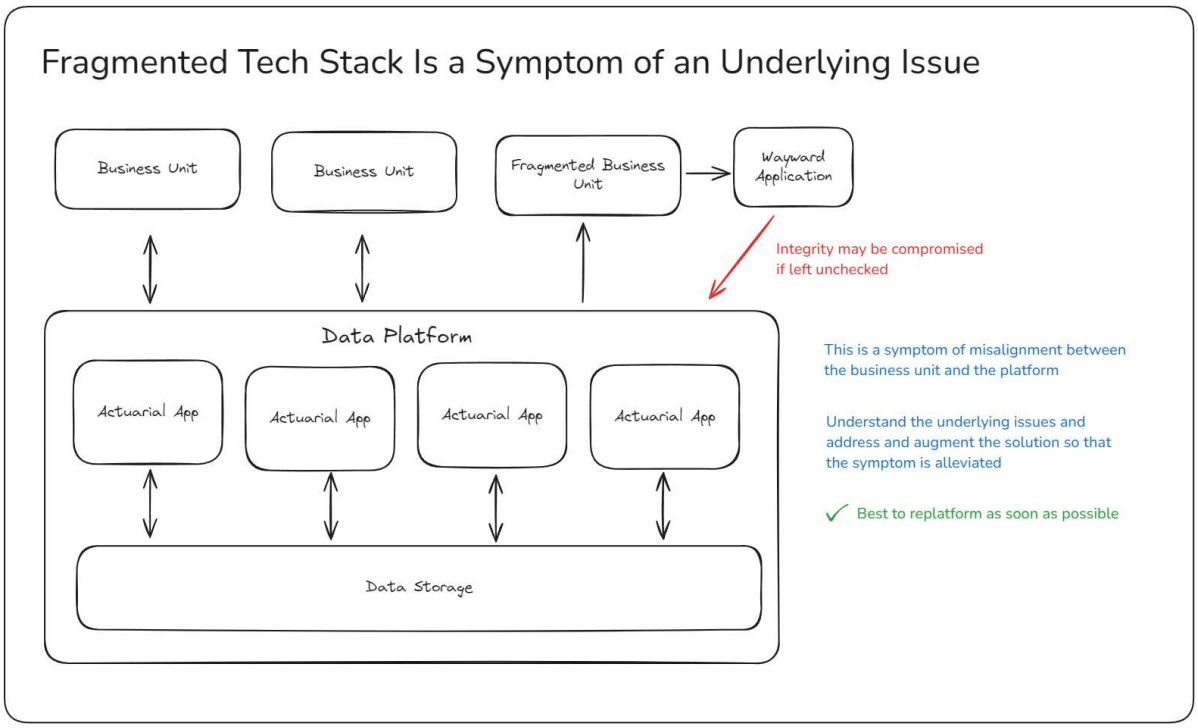
- **Maintenance Engineers (for Actuarial Applications)**: Production-grade actuarial systems need ongoing care — not just in terms of code correctness, but also **performance, reliability, and integration**. Maintenance engineers ensure that new model changes add newly requested features with scalability in mind. They have to ensure that pipelines don't break with upstream changes, that scalability is maintained (e.g., Spark jobs run efficiently), and that dependencies are actively monitored. This role might be part actuarial analyst, part software engineer, and part application owner.
- **Hybrid Product Owners or Tech Leads**: In some organisations, the gap between actuarial and technical teams is bridged by product owners or leads with hybrid experience. These individuals translate business requirements into technical deliverables, prioritise backlogs, and ensure alignment between actuarial goals and engineering execution.
- **Data Owners**: With increased data usage comes increased responsibility. Specific individuals or teams must be accountable for the accuracy, completeness, and timeliness of data flowing into actuarial models. This ownership is essential to support regulatory reporting, auditability, and operational trust.
- **Data Governance Leads**: As actuarial models consume more data from enterprise systems, the need for robust governance increases. Data must be classified, catalogued, and secured — especially when it includes personal or sensitive policyholder information. Data governance leads help define access rules, ensure lineage tracking, and enforce compliance across platforms.

Establishing these roles isn't just an HR exercise — it's **fundamental to long-term sustainability**. Without clearly owned responsibilities, production actuarial applications will become fragile, opaque, and prone to failure. As with all transformation, **roles must be designed, communicated, and staffed with intent** — otherwise, the burden of change will fall unevenly, and delivery will suffer.

5.3 TECH STACK FRAGMENTATION

Even with a clear strategy to modernise through a centralised data platform, **fragmentation of tooling and approaches** often emerges as a byproduct of transformation.

In practice, while a unified actuarial delivery platform is being designed and implemented, **splinter groups may spin up ad hoc or tactical solutions** to meet immediate business needs. This is rarely malicious — it's a reflection of the pressure to deliver short-term value, respond to reporting deadlines, or experiment with models in parallel to centralised efforts.



However, the side effect is **tech stack sprawl**: multiple versions of the same logic implemented in different languages, siloed scripts running outside governance controls, or lightly documented "quick fixes" that persist far longer than intended. Over time, this creates a growing delta between the "official" platform and the reality on the ground.

Left unmanaged, this can lead to:

- Conflicting logic between environments (e.g., different results from model in Excel vs model in Spark)
- Duplication of effort and inconsistent assumptions
- Security and compliance risks from unsupported or unreviewed code
- Technical debt that's hard to unwind

The root issue is often a mismatch between **business tempo and platform delivery cadence**. Business leaders need results today; platforms may take quarters to deliver. The solution isn't to eliminate tactical work, but to **acknowledge it, track it, and plan for convergence**.

Clear governance can prevent fragmentation from undermining the broader transformation effort:

- When and how temporary tooling is approved
- How it is documented and reviewed
- How and when it is absorbed into the centralised platform

These principles must be implemented and enforced as organisation-wide policies to foster greater unity and a more singular vision towards a scalable, transparent, and future-ready actuarial ecosystem.

In short, **modernisation efforts must balance long-term architecture with near-term agility** — or risk being undermined by the very business urgency and value creation they aim to support.

6 CONCLUSION

This paper has outlined the technical and organisational realities of modernising actuarial applications — a challenge that requires far more than just migrating models off Excel. It demands the ability to build scalable, auditable, and performant systems that integrate seamlessly with cloud-native data platforms. That means not only reengineering actuarial logic into production-grade code, but doing so with proper governance, version control, CI/CD, testing, and cross-functional alignment. These aren't just theoretical ideals — they are engineering disciplines that must be applied with precision.

For organisations looking to make this shift, the real differentiator is not just having the vision, but the technical ability to execute with clarity — to bridge actuarial depth with modern software practices.

7 REFERENCES

- https://www.pwc.com/gx/en/industries/financial-services/insurance/global-actuarial-modernisation-survey-2023.html
- https://www.pwc.com/us/en/industries/financial-services/library/future-of-actuarial-modeling.html
- https://www.milliman.com/en/insight/actuarial-model-governance-empowering-people-with-technology
- https://www2.deloitte.com/content/dam/Deloitte/us/Documents/financial-services/us-modernizing-the-insurance-actuarial-operating-model.pdf