CIO-to-CIO Talking Points: Architecture & Cost Stability

A CIO-Focused Conversation Guide on Cost Predictability and Calculating TCO

Prepared by Vimo

A Brief Overview

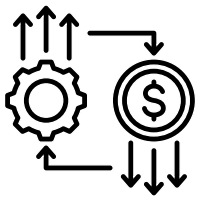
The biggest budget risk isn't initial pricing. It's total cost of ownership (TCO).

As policy changes, security expectations, and service demands evolve, the true cost of a system is no longer defined by what it costs to buy, but by what it costs to change. Systems built incrementally, heavily customized, or stitched together with bolt-on tools may appear cost-effective at first. However, over time, these approaches tend to create integration friction, vendor coordination complexity, and the need for unplanned procurements that drive volatility into state budgets.

A predictable, stable TCO has a lot to do with system architecture.

With the right architectural approach and vendor relationship, states can more accurately predict long-term TCO and enjoy the benefits of a more stable, reliable budget. Emergency consulting spikes and last-minute funding requests can be avoided, and operational overhead can be reduced. Agencies can start planning responses to future changes, instead of being caught up in a stressful cycle reacting to them. That's how cost becomes predictable again, and predictability is exactly what we want in our budgets.

Quick Talking Points

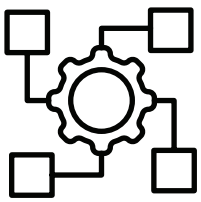


TCO is driven just as much by change as it is by licensing.

"Part of determining TCO is asking how easily the system can adapt without needing new contracts."

- Initial pricing is usually clear; how costs change over time is harder to predict.
- Policy, security, and service changes often pose stress on systems.
- Requiring procurements or consultants for routine updates inflates costs.

The real budget risk isn't what systems cost today – it's the cost of changing them.



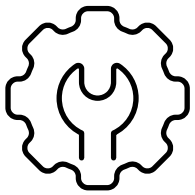
Attention to system design early on is key to achieving cost predictability.

"Budget cycles expose architectural limits and complexities over time."

- Heavily customized systems often complicate future changes.
- Integration-heavy architectures accumulate maintenance overhead.
- Rigidity forces states into reactive spending instead of planned investment.

Cost predictability is a function of design maturity as much as procurement discipline.

Quick Talking Points *(continued)*

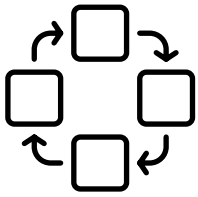


Separate tools seem cost-effective until you try to change something.

"Procurement and integration complexity almost always mean unplanned costs."

- Bolt-on database, security, and deployment technologies increase overhead.
- Every tool procured introduces risk related to integration and maintenance.
- When tools are loosely integrated, update demands cascade across contracts.

What appears cost-effective on paper can easily become fragmented in practice.

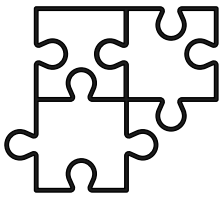


SaaS is both a pricing model and a life-cycle management model.

"With true SaaS, compliance and updates are part of predictable vendor operations."

- Vendor-managed infrastructure reduces surprise upgrade cycles.
- Security patches and compliance updates become part of routine maintenance.
- Multi-tenant enhancement cycles distribute innovation costs across clients.

When life-cycle responsibility shifts to the platform owner, cost volatility declines.



Fragmentation multiplies integration coordination and oversight burden.

"Every integration requires resources and stands to pose complications."

- Point-to-point integrations create break/fix exposure during policy shifts.
- Separate support agreements increase administrative overhead.
- Vendor coordination slows decision-making and amendment cycles.

Operational friction quietly drives up the total cost of ownership.

Predictable operations reduce hidden labor costs.

"Total cost of ownership includes staff time, not just vendor invoices."

- Manual deployments and testing increase over time.
- Fragile integrations tend to slow releases and extend outages.
- Burnout-driven turnover adds indirect replacement costs.

Stable architecture helps create stable teams, which in turn helps create stable budgets.

Questions to Invite Further Dialogue

- "If we had to implement a major policy change next quarter, would our architecture absorb it, or would it trigger new contracts and spending?"
- "Who owns the cost of change in our current model: us or the provider?"
- "How much of our TCO is driven by staff time and integration complexity?"