JOURNAL

ISSUE 138 – Q3 2026

Efficiency

TECH AND ACTIVITIES – AUTOMATION, INTELLIGENCE, SUPPLY CHAIN MANAGEMENT, CLOUD-BASED WORKFLOWS

This issue

Next Generation Talent

Enterprise AV

Content Chain

and much more

DON'T FORGET YOUR FIFA WORLD CUP IBC SPECIAL EDITION SUPPLEMENT

iamt™

connect | support | inform

Trusted by the industry. Driven by members.

Efficiency

viaccess·orca

The Cost Streaming Operators Can't See, Until They Measure It

Most streaming operators can calculate what they spend on content, infrastructure, and delivery separately. Almost none can quantify the cost of running all three together.

Benoit Brieussel
Vice President of Product & Solutions, Viaccess-Orca

Complexity is the hidden cost of systems procured, built, or deployed as standalone investments, then expected to operate as part of a connected service. Because that cost is spread across teams, workflows, and technology layers, it rarely appears on any single budget. That absence is why complexity remains one of the least examined and most persistent threats to streaming margin.

The industry's cost pressures are well documented individually: content licensing, infrastructure spend, regulatory compliance. What is less examined is the cost that sits between those categories rather than inside any one of them. As streaming platforms have grown to serve more devices, more markets, and more vendors within a single stack, the cost of coordinating that growth has expanded in step with it, and largely without a corresponding line of accountability.

Where Complexity Accumulates, and Why It Stays Invisible

Device fragmentation is the clearest example. Every new feature, version, or platform update must be deployed and certified separately across a fragmented device landscape – Samsung's Tizen, LG's webOS, Android TV, Apple's tvOS, Roku's BrightScript environment, and Amazon Fire TV among the major platforms – each with its own SDK, testing matrix, and certification pipeline. This is a fixed cost of doing business for any TV service provider, and each product iteration adds to it.

Multi-vendor architecture compounds the same problem from a different angle. Integration costs between vendors are routinely budgeted as one-time project expenses, but in practice they recur: a change in one system produces side effects in others, and identifying and resolving those interactions under the time pressure of a live event or a major content launch is both expensive and high-risk. Across large operator deployments, those seemingly contained dependencies can become recurring obligations. A third-party or partner technology library integrated at build time, for example, may require repeated version updates as the library evolves or defects emerge. An application built around a video streaming player component can create similar downstream work as post-launch requirements expand, whether to support low-latency delivery for live programming, thumbnail generation for VOD, or faster channel-change performance improvements. This cost scales with the stack itself.

Duplicated infrastructure adds a third layer. Operators running multiple brands, regions, or service tiers on separate infrastructure absorb a compounding duplication cost: redundant engineering resources, redundant infrastructure instances, and the overhead of propagating every fix and upgrade across parallel systems.

Each of these costs is structural rather than incidental. Operators measure cost by department and supplier because that is how organizations are built, with each team owning its scope and reporting against its own budget.

Complexity cost is generated at the seams between those departments and suppliers, where no team holds full ownership and no metric captures the full picture. The result is a category of cost that is real, growing, and largely unmanaged, a consequence of accounting structures built around departmental scope instead of platform-wide interaction.

Making Complexity Visible

One of the few metrics precise enough to surface this complexity is cost per viewer, the total platform operating cost divided by active viewers in a given period. Because it aggregates infrastructure, delivery, operational overhead, and acquisition spend into a single figure rather than leaving them siloed by department, it forces complexity's distributed costs into a shared frame. A spike in device certification cost, a recurring integration failure, a redundant infrastructure instance carried across three regional deployments – none of these shows up cleanly in a departmental budget, but each moves cost per viewer, and the metric does not care which team's line item it came from.

This matters because complexity cost, left siloed, is nearly impossible to prioritize against other spending decisions. Aggregated into a single per-viewer figure, it becomes comparable to content spend, infrastructure investment, and every other cost category competing for the same budget. That comparability is what turns complexity from an accepted cost of doing business into a decision operators can actually act on.

Reducing Complexity Without Disruption

The most effective response to accumulated complexity is rarely wholesale replacement. For operators running legacy on-premises infrastructure, that means full cloud-native migration usually isn't the right first move. What's often called modernization, or cloudification

elsewhere in the market, wraps a Kubernetes-based microservices layer around existing monolithic systems. This method delivers the scalability and modularity benefits of cloud-native architecture without the disruption and capital expenditure of replacing the stack outright, and it avoids a spike in platform cost precisely when margin pressure is highest.

Consolidation addresses the duplication problem directly. Moving multiple brands, regions, or tiers onto a unified backend, while preserving differentiated front-end experiences where the business requires it, eliminates redundant engineering and infrastructure overhead without forcing a single undifferentiated product across markets that need to remain distinct.

Automation reduces the operational surface each release touches, and by extension the number of places complexity can compound. Deployment automation through standardized templates and CI/CD toolchains lowers the engineering overhead of each release and reduces the risk of the cross-system side effects that make multi-vendor architectures expensive to maintain. AI applied across the software development lifecycle compresses the time and headcount required to keep platform components current, while shifting quality operations from reactive to proactive.

The Discipline This Requires

Streaming complexity is not going to decline. Device ecosystems will keep fragmenting, vendor stacks will keep growing, and regulatory obligations will keep adding engineering requirements on top of both. Operators that have built the visibility to see where complexity accumulates, and have the discipline to treat it as a managed cost rather than an absorbed one, will be well-positioned for that trajectory. Cost per viewer, and metrics built on the same aggregating logic, are what make that discipline possible. As complexity continues to compound, that visibility will increasingly keep operators that manage their margin deliberately from wondering where it went.