STACK, the operating model for enterprise services at scale

By Rahul Jindal

STACK is a five-pillar operating model for enterprise services. Strategy, Transformation, Automation, Content, Knowledge. Five letters, one for each pillar, in the order they tend to fail.

Strategy sets direction. What capabilities to build. What to sunset. How to allocate resources across the other four pillars. Without it, the other pillars optimize locally and drift globally. The diagnostic question for Strategy is sharp. Does your services org have a strategy that is distinct from the company strategy, or does it just execute whatever comes in.

Transformation is how the function changes itself. Most services orgs treat transformation as a project. Five-year plan, milestones, a Gantt chart. The orgs that actually transform treat it as a permanent muscle. There is always a portfolio of moves in flight, with named owners and named retirement criteria.

Automation is where work gets removed, not just sped up. The trap is automating yesterday's workflow. The orgs that win redesign first, then automate. The ones that lose paste a copilot on top of the existing process and call it transformation.

Content is the most neglected pillar by a wide margin. Every services organization produces enormous amounts of knowledge: policies, procedures, training materials, case resolutions, FAQs. In most orgs, this content is scattered across wikis, shared drives, and people's heads. It degrades over time. Nobody owns it. The diagnostic question is brutal. If a new employee joined your team today, could they find every policy and procedure they need within a day, or would they spend weeks asking around.

Knowledge is the meta-pillar: how the organization learns from its own work and re-distributes the learning. Most orgs do this implicitly, through tenure and apprenticeship.

The orgs that scale do it explicitly, with named curation roles and structured knowledge cycles.

STACK is built on the observation that most enterprise services orgs are running with two pillars partially functional and three either missing or improvised. Naming the five lets you see which one is your binding constraint. Working on a non-binding pillar feels productive and changes nothing.

The full paper walks each pillar with its own diagnostic question, its own failure mode, and its own first move. Read it if you run a services function or a transformation office.
