Custom Code and Technical Debt: What Actually Drives S/4HANA Migration Cost

Custom ABAP code and the technical debt built up around it are usually the biggest wildcard in an S/4HANA cost and timeline estimate. With SAP's December 2027 deadline fixed, getting an honest read on that code early is what separates a real budget from a guess, and AI-assisted analysis can help get there faster.

8 min read | Orpington Technologies Insights

SAP's mainstream maintenance for ECC ends December 31, 2027. That date alone would be reason enough to plan a migration, but the pressure doesn't stop there. As the installed base shifts toward S/4HANA, the pool of consultants, systems integrators, and third-party tools built around ECC-era SAP thins out too. Waiting doesn't just risk running an unsupported system. It risks doing so with fewer people left who know how to support it.

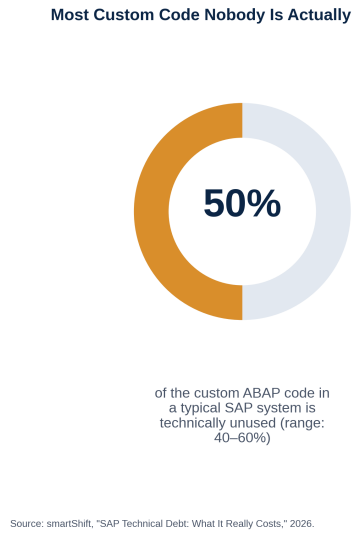
For most organizations, though, the deadline isn't the hard part of the decision. The hard part is what's sitting inside their own systems: custom ABAP code accumulated over ten, fifteen, twenty years, much of it undocumented, built by developers who've since moved on. Before anyone can price out a migration or commit to a timeline, someone has to figure out what that code actually does, what's still load-bearing, and what can be safely left behind. Get that assessment wrong, or start it too late, and it becomes the thing that blows up both the budget and the schedule. AI-assisted tools are increasingly part of how teams get through that assessment faster, but the underlying work, and the risk of getting it wrong, is the real subject here.

The Scale of What's Actually in There

smartShift's analysis of typical SAP ECC environments puts the average system at roughly 22,000 custom objects and 2.7 million lines of executable custom code, carrying an estimated 300,000 individual technical-debt issues spanning performance, security, and stability concerns. None of that means custom code itself was a mistake. smartShift's own research found that 95% of organizations run

custom code specifically because it supports real business processes SAP's standard configuration doesn't cover.

The problem is the other half of that picture: 40–60% of the custom code sitting in a typical system is technically unused. Nobody has deleted it. In some cases nobody is fully certain it's safe to. smartShift estimates the ongoing cost of simply carrying that unused code at $0.25 to $0.63 per line per year. It's a small number multiplied across millions of lines, and the bill comes due again, at a larger scale, the moment a migration project has to decide what to do with every one of those objects.



Share of custom ABAP code in a typical SAP system that is no longer actually executed.

Why the Assessment Itself Can't Drag On

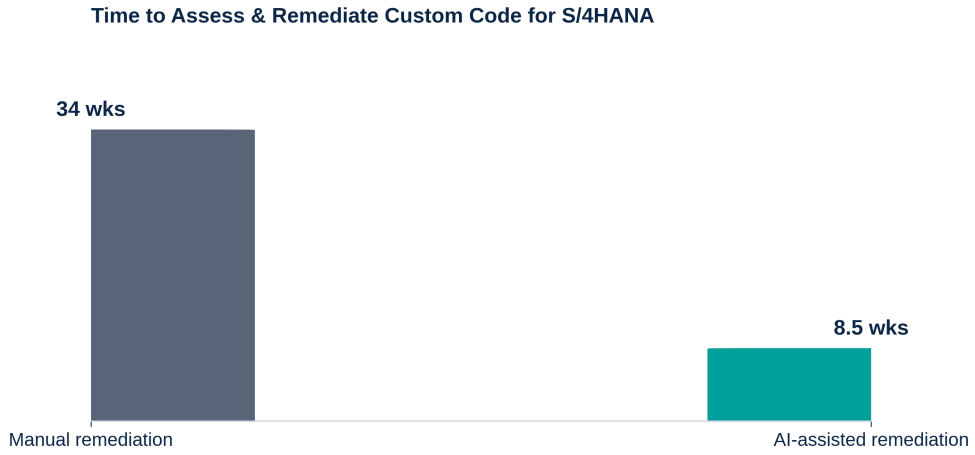
Custom code assessment has traditionally meant static analysis tools paired with manual developer review: reading through enhancement after enhancement to determine whether it's still called, what it depends on, and whether the underlying business logic survives the move to S/4HANA's simplified data model. On a system carrying 22,000 custom objects, that's not a task a small team finishes in a sprint. smartShift puts the traditional timeline at six to ten months, and that's before remediation or testing even starts.

The risk isn't only time. Reviewers who weren't around when the code was written have to reconstruct intent from the code itself, with little documentation to lean on, and a wrong call in either direction is costly. Retire something that's still load-bearing and it breaks in production. Keep too much "to be safe" and the technical debt just moves into the new system. On a fixed 2027 deadline, an assessment that drags compresses everything scheduled after it: design, build, and testing all inherit the delay.

AI-assisted tools have made a real, if secondary, difference here. smartShift reports that AI-assisted analysis compresses the traditional six-to-ten-month assessment down to a matter of weeks for teams that use it, and SAP's own Joule tooling shows comparable gains elsewhere in the ABAP and BTP development workflow, according to SAVIC Technologies' benchmarking.

59%

of SAP customers cite extensive testing as a top challenge in custom code migration. That burden scales directly with how much undocumented custom code a system carries. (smartShift)



Source: smartShift, 2026 — reported as "6–10 months" manual vs. "weeks (~4x faster)" AI-assisted; chart shows range midpoints converted to weeks for comparison.

What Still Requires a Person to Decide

AI-assisted tools are good at answering one question: is this code still called, and by what. They're far less reliable on a second one: does the business still need what this code does. That's a judgment call about process, not syntax. Take a custom pricing routine that hasn't fired in eighteen months. It might be genuinely dead, or it might exist for a customer segment that only transacts once a year. Getting that distinction wrong in either direction has real cost. Delete active logic and something breaks in production. Keep everything "to be safe" and you've just recreated the technical debt problem inside the new system.

The organizations getting the most reliable outcomes from AI-assisted code assessment pair it with structured business-process owner sign-off on every object flagged for retirement. The AI output gets used to focus a scarce group of experienced reviewers on a manageable shortlist, rather than asking them to review 22,000 objects from scratch, or trusting an automated recommendation with no review at all.

Where This Leaves You

Custom code assessment is where a lot of migration budgets go wrong early, and where an outside technical review earns its keep fast. It's hard for the team that wrote the code to be fully objective about what's safe to retire, and a rushed or overly cautious call here sets the tone for everything that follows. Orpington Technologies leads S/4HANA migrations built around exactly this kind of independent code and technical-debt assessment, whether as a standalone review or as part of the broader project.

AI-assisted analysis is one tool the team uses to shorten a workstream that has historically run long, alongside deep SAP delivery experience and a track record of migrations completed ahead of the 2027 deadline. Learn more about Orpington's approach at orpingtontech.com.

Sources & Further Reading

[1] smartShift — SAP Technical Debt: What It Really Costs and How to Measure It in Custom Code

[2] SAVIC Technologies — SAP AI ROI: Real Numbers, Enterprise Reality Check, 2026

[3] Basis Technologies — The True State of S/4HANA 2025