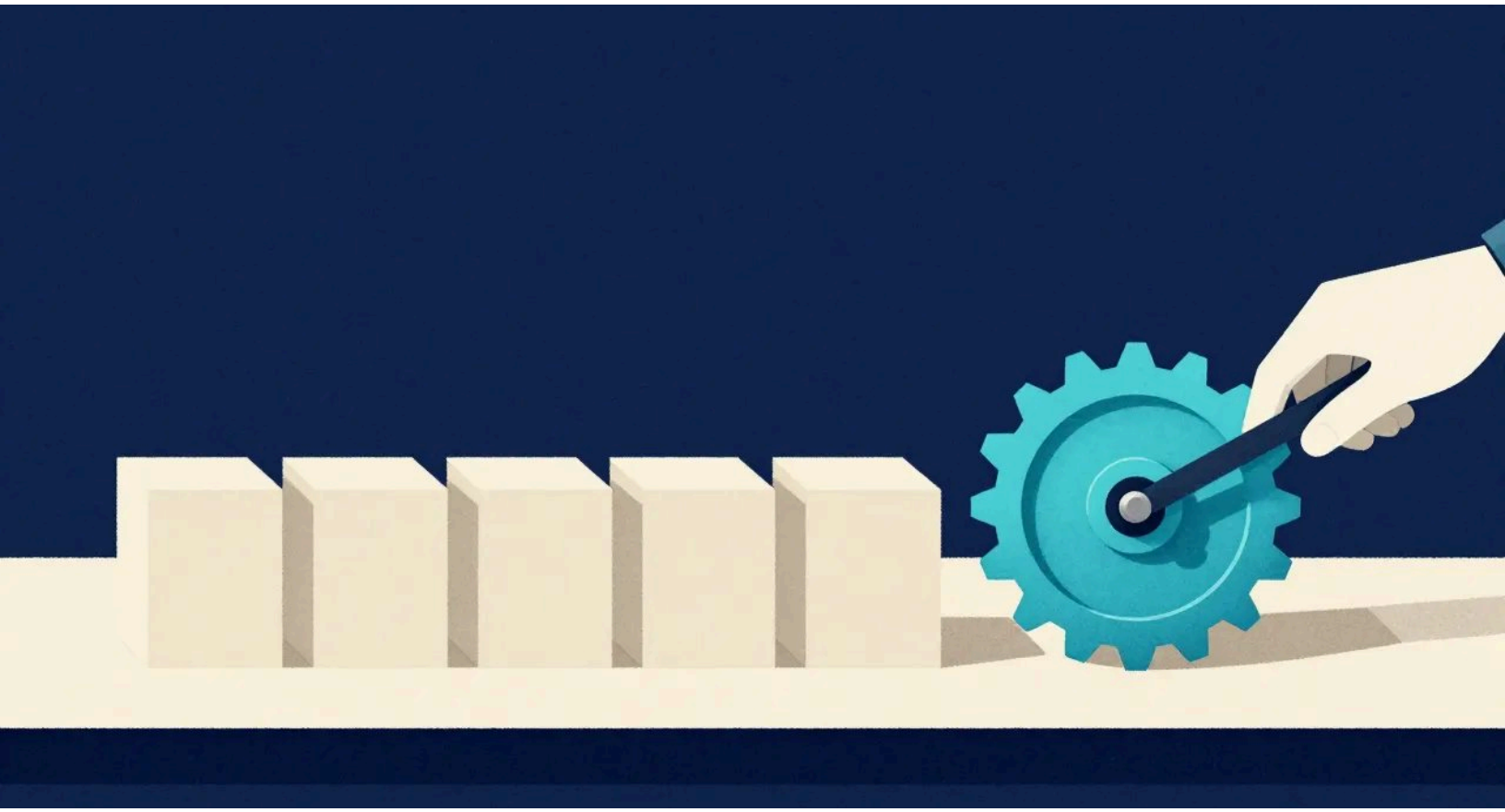


EXECUTIVE PAPER · LEGACY SYSTEMS

Buy the Commodity, Build the Differentiator

Most software budgets go to systems that cannot set a company apart. Executives who sort their portfolio first spend less on upkeep and more where customers notice.

AUTHOR

Chris Watkins

SERVICE

Legacy Systems

FOR

CEO, CFO, CTO · 11 min read

AT A GLANCE

The problem. Companies keep paying to customize software that does the same job at every competitor: general ledger, payroll, email, ticketing. Each change to a packaged product becomes code someone must retest and repair at every upgrade, for as long as the product runs. Public ERP programs in 2024 to 2026 show customization turning single-digit-million projects into nine-figure ones.

Why it persists. Departments ask the software to match how they work today, and saying yes looks cheaper than retraining. The maintenance bill arrives years later in someone else's budget. AI coding tools now make custom work look nearly free to start, which tempts teams to build even more of it.

The recommendation. Sort every system into commodity or differentiator with one test: would a customer choose you, or pay you more, because of how you do this? Buy the commodity, adopt the vendor's process, and keep custom code outside its core. Put your engineering budget, and your AI tooling, into the few systems that pass the test.

The Decision in Front of You

Every company runs two kinds of software. One kind does work that every competent company does the same way: paying people, closing the books, sending email, tracking support tickets. The other kind does the work customers pay for: the pricing engine, the routing model, the product itself.

The decision is whether to treat these differently in budget and governance. Most organizations do not. They apply the same appetite for customization to payroll as to the product, and they fund both from one project pot.

The stakes are large because software projects already miss badly on average. A McKinsey study with the University of Oxford of more than 5,400 IT projects, published in 2012, found large projects ran 45% over budget and delivered 56% less value than predicted, and 17% went badly enough to threaten the company [2]. The data is old, and the pattern in this paper's recent cases suggests it still holds.

What's Happening

Customization is where ERP programs break

West Sussex County Council. The council approved £2.6 million in November 2019 to replace its SAP system with Oracle Fusion [4]. By March 2026 the estimate stood at about £41 million: £14.07 million already spent plus £27.05 million approved to finish, roughly 15 times the original figure [4]. The council ended its contract with its systems integrator in 2023, after finding no real prospect of a deliverable

plan, and is selling buildings to pay for the work. HR and payroll for more than 20,000 users moved to October 2026 [4]. Its auditor, EY, reported weaknesses in budgeting, governance and risk management [4]. The lesson is about scale: a commodity back-office replacement became one of the largest capital commitments in the council's portfolio.

Zimmer Biomet. In September 2025 the medical device maker sued Deloitte Consulting for at least $172 million over an SAP S/4HANA program [5]. The complaint alleges Deloitte promised delivery with minimal customization, then delivered an overly customized system with defects and functionality gaps after a July 2024 go-live, on $94 million in fees, followed by tens of millions of dollars in remediation and supply chain disruption [5]. Deloitte calls the claim meritless and has moved to dismiss [6]. The allegations are unproven. They are still instructive, because both sides of the dispute treat minimal customization as the standard a competent ERP program should meet.

Birmingham City Council is the best-known example, covered in depth elsewhere. One detail belongs here: the council planned to run Oracle out of the box, built customizations anyway, and a custom banking reconciliation failed, leaving it unable to produce auditable accounts. The forecast cost reached £144.4 million against an initial £19 million [3].

Upgrades turn old customizations into new bills

Packaged software changes on the vendor's schedule. SAP ends mainstream maintenance for Business Suite 7, which includes ECC 6.0, at the end of 2027, with extended maintenance to 2030 at a premium [7]. As of September 2024, Gartner found only 37% of ECC customers had licensed S/4HANA [7]. Every custom object a company added to ECC over twenty years now has to be found, judged, and rewritten or retired before that move. The companies with the most customization face the largest bill and the least room in the timeline.

The commodity is getting more expensive to rent

Buying has its own rising cost. Vertice, which tracks more than $75 billion of processed software spend, measured SaaS price inflation at a record 16.4% in June 2026 and a preliminary 14.3% in September 2026 [8]. Gartner forecast software spending would grow 14.2% in 2025 to $1.25 trillion, and its analyst said much of that growth would only offset price increases on recurring contracts [9].

A $1 million annual SaaS contract rising 12% a year costs about $1.57 million in its fifth year and $6.35 million over five years, against $5 million at flat pricing (author's arithmetic). That cost is real, and the answer to it is a hard negotiation and a tested exit path. Rebuilding payroll in-house would trade a 12% price rise for the full cost of owning payroll code.

AI makes custom code cheaper to write, and the upkeep remains

The newest objection to buying is that AI coding assistants make building nearly free. The evidence so far is mixed and argues for caution.

METR ran a randomized trial in early 2025 with 16 experienced developers on 246 real tasks in mature codebases they knew well. With AI tools allowed, tasks took 19% longer, while the developers believed they had been about 20% faster [10]. METR's February 2026 follow-up found point estimates of an 18% speedup for returning developers and 4% for new ones, with intervals that include a slowdown. METR itself called the data weak evidence, because many developers refused to work without AI [11].

Google's DORA research reached a similar split at the organizational level. Its 2024 report associated rising AI adoption with an estimated 1.5% fall in delivery throughput and a 7.2% fall in delivery stability [14]. Its 2025 report, built on nearly 5,000 respondents, found AI adoption now associated with higher throughput while still associated with lower stability [12]. GitClear's analysis of 211 million changed lines from 2020 to 2024 found copy-pasted lines rising from 8.3% to 12.3% and refactoring falling from 25% of changed lines to under 10% [13].

Read together, AI lowers the cost of producing a first version. It has not been shown to lower the cost of owning code for ten years, and some evidence points the other way. Every line an assistant writes inside a vendor's ERP is a line your team must carry through the next upgrade.

Why the Usual Approach Fails

The common mistake is letting the department that will use a system decide how much to change it. Finance wants the ledger to match last year's chart of accounts and approval chain. HR wants the old leave rules. Each request is reasonable and cheap on its own. The project team, measured on user acceptance, says yes.

The cost lands later and elsewhere. Custom code in a packaged product must be retested at every vendor release, rewritten at every major version, and understood by someone after the original developer leaves. It rarely has an owner or a budget line after go-live. The incentive stays the same from project to project: the people who ask for customization never see its maintenance bill.

There is a second mistake in the other direction. Companies buy a package for something that does set them apart, then bend their best process to fit the vendor's average one. A distributor whose edge is same-day allocation across forty sites gains nothing from a generic allocation module. That company should build the allocation engine and buy the ledger.

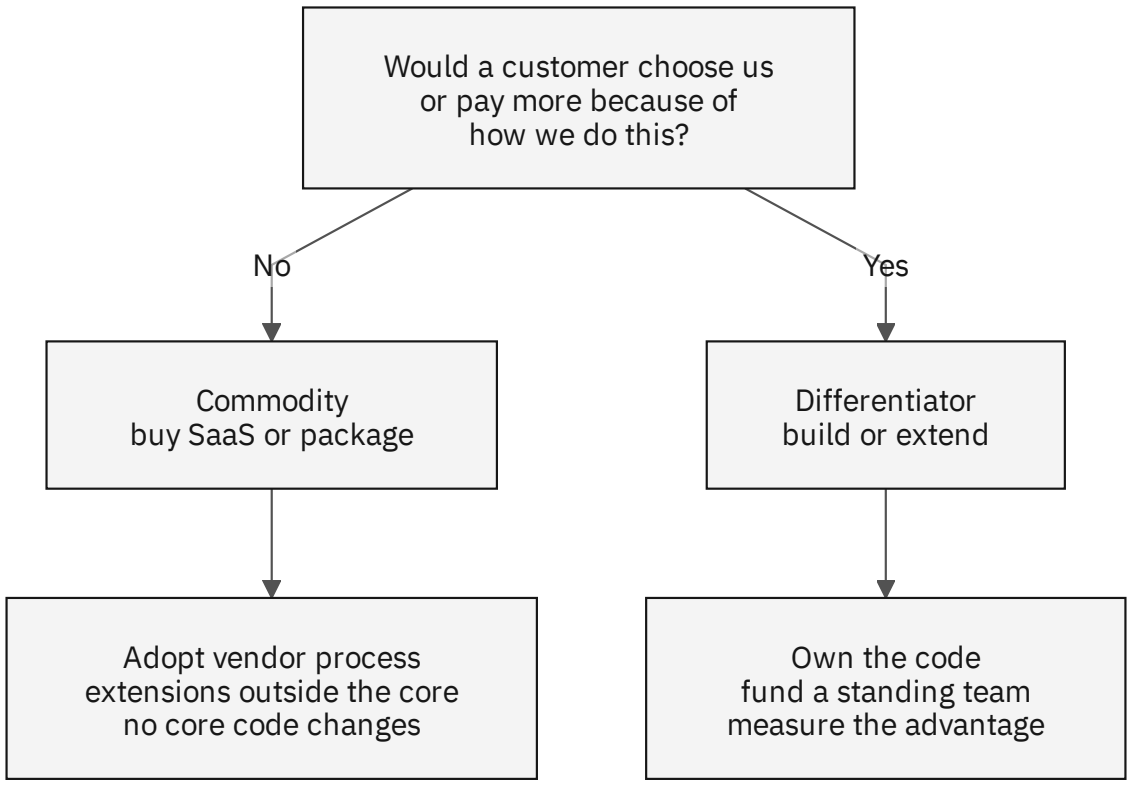
Both mistakes come from skipping the classification step. Without it, build-or-buy gets decided project by project, by whoever is in the room.

The Framework: The Customer Test

Two established frameworks make the same split. Simon Wardley's mapping places every component on an evolution axis from genesis through custom-built and product to commodity, and recommends differentiating on the evolving components while sourcing the commodity ones externally [1]. Gartner's

pace-layered model sorts applications into systems of record, differentiation and innovation, each with its own rate of change and governance [15]. An executive team can apply the same idea with one question asked of every system.

Exhibit 1. The Customer Test sorts each system into one of two funding paths



Source: author's framework, drawing on Wardley mapping [1] and Gartner pace layering [15].

Commodity systems get bought and configured. The default is the vendor's process. A change to the vendor's core code needs sign-off from the CFO, because it creates a permanent liability. Needs the package cannot meet go into separate, small extensions that talk to it through published interfaces, so an upgrade does not break them. Contracts include data export and an exit plan, given the pricing trend above.

Differentiating systems get built or deeply extended, owned by a standing team, and measured against the advantage they are supposed to create: margin, conversion, delivery time. This is where AI coding tools belong, because the code is yours to keep and improve anyway, and the team that owns it will review what the tools write.

Most portfolios sort fast. In the author's experience, a mid-size company finds that only a handful of its systems pass the Customer Test.

What customization costs over five years

Exhibit 2. Illustrative five-year cost of customizing a commodity ERP versus configuring it

Assumption or result	Heavy customization	Configure, extend outside core
Custom build cost at implementation	$2.0M	$0.5M
Annual maintenance of custom code (20% of build)	$400K	$100K
Upgrade retest and repair, per year	$150K	$50K
Five-year cost of custom work	$4.75M	$1.25M
Difference		$3.5M

Source: author's illustrative estimate. Assumes a mid-size company, annual maintenance at 20% of build cost, and one major vendor upgrade every two years. License cost is the same in both columns and excluded. Substitute your own figures.

The model leaves out the larger risk in the recent cases, a failed go-live that halts billing, payroll or shipping.

Costs, Risks, and Tradeoffs

Exhibit 3. Tradeoffs of buying the commodity and building the differentiator

Area	Cost or risk	Mitigation
Process change	Staff must learn the vendor's way of doing payroll or procurement; productivity dips for a quarter or two	Budget training and a short dual-running period; publish the reasons for each change
Vendor pricing	SaaS prices rose double digits in 2026 [8]; lock-in weakens your hand at renewal	Multi-year price caps, data export clauses, a tested exit plan for each critical vendor
Lost fit	Some local rules get handled by workaround or manual step	Allow extensions outside the core; review workarounds yearly and retire the ones nobody uses
Build capacity	Differentiators need a standing team, which costs more than a project	Fund fewer differentiators, fully; stop the ones that fail to show their advantage
AI-built code	Faster first versions can hide rising instability and duplication [12][13]	Hold AI-written code to the same review, testing and ownership rules as any other
When this is wrong	Regulated or unusual operations where no package fits (some utilities, defense, niche finance); or a vendor market so concentrated that renting is the larger risk	Build the core deliberately, with a standing team and the same lifetime cost discipline

Source: author's analysis.

How to Act

Exhibit 4. A one-year roadmap

Window	Actions	Owner	Milestone
First 30 days	List every business system. Apply the Customer Test to each with the business owner present. Record the count of custom objects in each commodity system.	CTO with CFO and business unit leaders	Every system classified; customization inventory for the top five commodity systems
First 30 days	Freeze new core customization of commodity systems without CFO sign-off.	CFO	Sign-off rule in force; first requests reviewed
First 90 days	For each commodity system, sort custom objects into retire, move outside the core, or keep with a written reason. Start with any system facing a vendor deadline, such as SAP ECC in 2027 [7].	Application owners	Plan covering every custom object in the top five systems
First 90 days	Review SaaS renewals due in the next 18 months for price caps and exit terms.	Procurement with IT	Price cap or exit clause negotiated on each renewal that comes due
First 90 days	Name a standing team and an advantage metric for each differentiating system.	CEO with product leaders	Each differentiator has a team, a budget and one measured outcome
First 365 days	Retire or move out at least half of the custom objects marked for it. Report custom code counts and upgrade cost to leadership each quarter.	CTO	Custom objects in commodity systems down 50%; next vendor upgrade completed with no core code changes

Source: author's recommendation.

Questions to Ask Your Team

1. Which of our systems would a customer notice if we ran them exactly like our competitors do?
2. How many custom objects sit inside our ERP, HR and CRM systems, and who owns each one?
3. What did our last major upgrade cost, and how much of that was retesting customizations?
4. Which vendor support deadlines fall in the next three years, and how much custom code stands in the way?
5. Where are we spending engineering time on a commodity system that a configuration change or an off-the-shelf add-on would cover?
6. For each system we built ourselves, what advantage does it create, and how do we measure it?

7. If our largest SaaS vendor raised its price 15% at renewal, what would we do?
8. Who reviews and owns code written with AI assistants after the person who prompted it moves on?

Closing

Software that does the same job at every company should cost as little to own as possible, which means buying it, adopting its process and keeping your code out of its core. Engineering money, and the new speed from AI tools, belongs in the few systems that make customers choose you. Classify the portfolio once, put a CFO signature on every core customization, and review the custom code count every quarter.

Sources

[1] Wardley Maps. "The Four Stages of Evolution." Passage from Simon Wardley's *Wardley Maps*. https://www.wardleymaps.com/book-passages/wardley-on-evolution.

[2] Bloch, M., Blumberg, S., and Laartz, J. "Delivering large-scale IT projects on time, on budget, and on value." McKinsey & Company, with the University of Oxford. 2012. https://www.mckinsey.com/capabilities/tech-and-ai/our-insights/delivering-large-scale-it-projects-on-time-on-budget-and-on-value.

[3] The Register. "Birmingham City Council's Oracle ERP fiasco now £144M and still not working." January 29, 2026. https://www.theregister.com/2026/01/29/birmingham_oracle_latest/.

[4] The Register. Report on West Sussex County Council's Oracle Fusion programme costs and delays. March 16, 2026. https://www.theregister.com/2026/03/16/west_sussex_oracle/.

[5] Loeb & Loeb. "Loeb Represents Zimmer Biomet in $172 Million Lawsuit Against Deloitte." Summary of complaint filed in New York Supreme Court, September 4, 2025. 2025. https://www.loeb.com/en/newsevents/news/2025/09/loeb-represents-zimmer-biomet-in-172-million-lawsuit-against-deloitte.

[6] MassDevice. Coverage of Deloitte's response and motion to dismiss in the Zimmer Biomet ERP lawsuit. 2025. https://www.massdevice.com/zimmer-biomet-deloitte-response-erp-lawsuit-looking-glass/.

[7] The Register. Report on SAP's ECC 2027 and 2030 maintenance deadlines and 2033 transition option, citing Gartner data on S/4HANA licensing. January 28, 2025. https://www.theregister.com/2025/01/28/sap_extends_support_deadline/.

[8] Vertice. "SaaS Inflation Rate." Monthly index from processed software spend. 2026. https://www.vertice.one/insights/saas-inflation-rate.

[9] Tech Monitor. Coverage of Gartner's January 2025 worldwide IT spending forecast. 2025. https://www.techmonitor.ai/digital-economy/it-spending-hit-5-6-trillion-2025-driven-by-software-ai.

[10] Becker, J., Rush, N., Barnes, E., and Rein, D. "Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity." METR. arXiv:2507.09089. 2025. https://arxiv.org/abs/2507.09089.

[11] METR. "We are Changing our Developer Productivity Experiment Design." February 24, 2026. https://metr.org/blog/2026-02-24-uplift-update/.

[12] Google Cloud. "Announcing the 2025 DORA Report: State of AI-assisted Software Development." 2025. https://cloud.google.com/blog/products/ai-machine-learning/announcing-the-2025-dora-report.

[13] GitClear. "AI Copilot Code Quality: 2025 Research." 211 million changed lines, 2020 to 2024. 2025. https://www.gitclear.com/ai_assistant_code_quality_2025_research.

[14] Google Cloud. "Announcing the 2024 DORA report." 2024. https://cloud.google.com/blog/products/devops-sre/announcing-the-2024-dora-report.

[15] Vanbrabant, F. "Pace layering an application portfolio." Description of Gartner's Pace-Layered Application Strategy. 2025. https://frederickvanbrabant.com/blog/2025-08-10-pace-layering-an-application-portfolio/.

The views in this paper are the author's own and do not represent any employer. Scenarios are hypothetical. This paper is not legal, financial, or compliance advice.