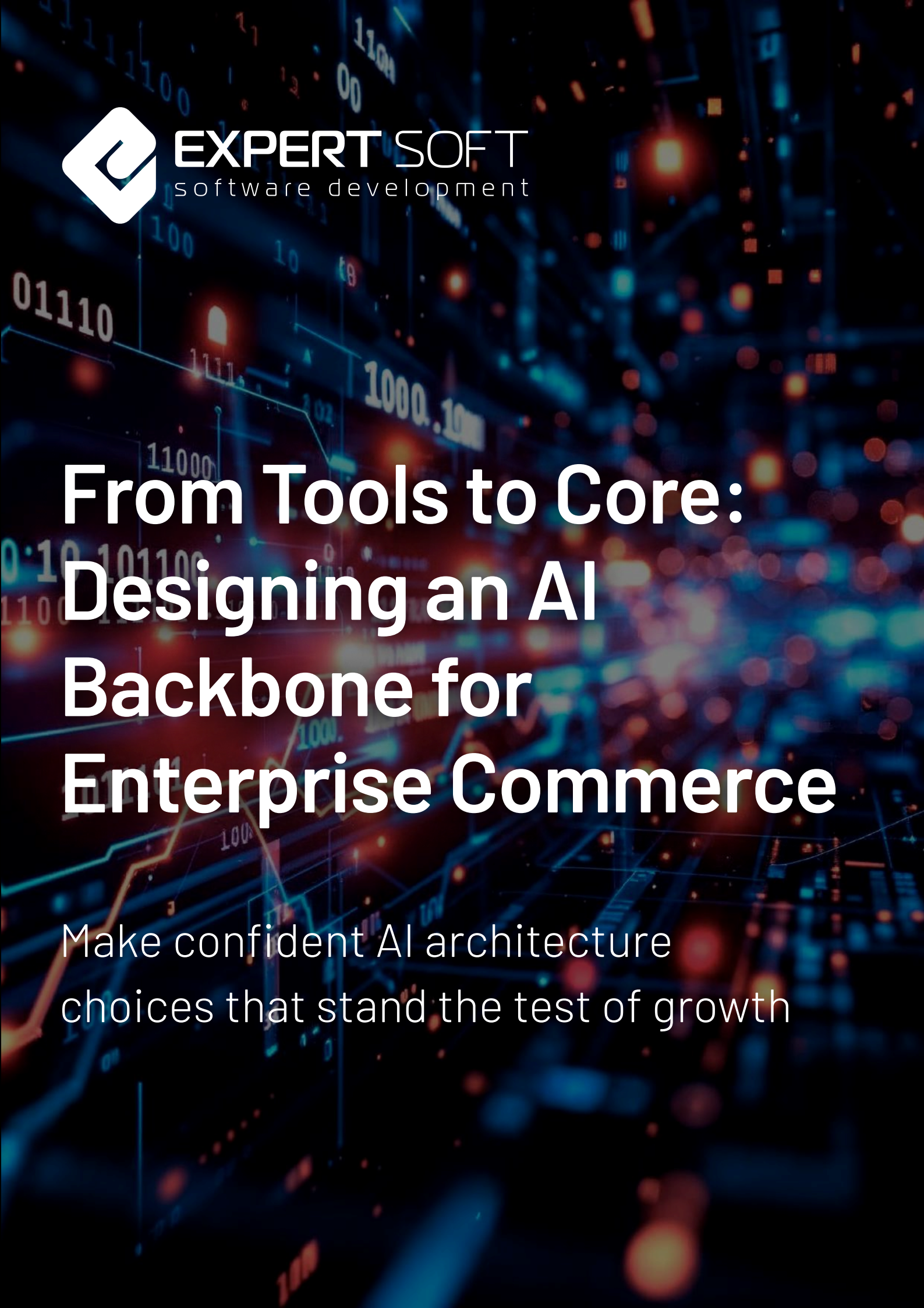


From Tools to Core: Designing an AI Backbone for Enterprise Commerce

Make confident AI architecture choices that stand the test of growth

When large companies decide to adopt AI, it's rarely about implementing a single solution for a single task. These organizations think long-term: they look beyond immediate wins and focus on long-term capabilities. That's why, for them, the real question isn't which AI tool to choose, but **how to establish an AI core** that can power many use cases over the next two to three years without constant rework.

If you ask the same question, this whitepaper will help you make that call with clarity. You'll explore three paths to building an AI backbone — Product-first, Platform-first, and Custom-first — and see how each behaves under real enterprise pressure: risk, cost, performance, and scale.

Along the way, you'll get practical thresholds, readiness markers, and 90-day playbooks to move from decision to delivery and build an AI core that keeps its promises long after the pilot phase ends.

Table of Contents

Three Roads to an AI Core and Where Each One Leads

When you move past chasing one-off AI tools and commit to building a real core, a new question appears: "How to build it?". The market offers plenty of options, but the decision isn't really about vendors or model specs. It starts with architecture. There are three ways to shape that backbone.

Product-first approach includes ready-made SaaS tools for well-defined problems, e.g., automated recommendations or personalization engines. It's the fastest way to prove value and the safest start when use cases are standardized. However, these tools solve today's problem efficiently but rarely stretch to tomorrow's needs.

Platform-first option takes it further. Teams assemble capabilities on broader AI platforms — like IBM ICA or Copilot Studio — to build internal tools fast and stay within governance. It gives you flexibility to experiment and orchestrate workflows, but you're still bound by the platform's primitives, connectors, and roadmap.

Custom-first is for enterprises that already know what differentiation looks like, where millisecond latency, deep ERP/SAP integration, or multi-market logic define success. It's not the fastest start, but it's the approach that scales cleanly as complexity rises.

There is a more detailed comparison

	PRODUCT-FIRST	PLATFORM-FIRST	CUSTOM-FIRST
Data contracts & schema control	narrow data shapes optimized for that product's feature set	prebuilt connectors and schemas; custom fields often end up as "notes/ other."	you own schemas, denormalization, freshness windows; easiest to reuse across use cases.
Latency SLOs at peak	CDN help is common, but inference queues are shared	shared tenancy, opaque queues; burst handling varies	tuned caches, precompute windows, warm pools sized to your peak profile
Experimentation cadence	A/B often limited to the product's UI layer	bound by exposed "knobs"	feature flags/canaries + offline eval sets enable continuous iteration
Multi-brand / multi-locale policy	typically one catalog flavor; locale expansion increases cost/complexity	risks of duplicated rules or global overrides	policy layer separates global logic from brand/ market specifics
Integration depth	shallow integrations unless you buy extra modules	API coverage varies; webhooks latency may pinch	direct adapters; near-real-time ingestion where it matters
Observability & explainability	black-box scoring with exports at best	dashboards, limited traceability	per-decision traces, replayable datasets, audit trails
Cost governance	tier jumps at usage peaks	coarse alerts; little route-level control	coarse alerts; little route-level control budget guards in code, route-level unit economics
Model neutrality	tied to a preferred model stack; no swap or routing, often requiring changing products	curated model list; swapping within a supported list is easy, beyond it needs extensions/ waiting	adapters for best-fit models per task/market
Vendor lock-in exposure	proprietary features/ schemas lock-in; exit often means re-implementing logic and migrating data.	API/DSL ties; export possible, but costly	lock-in moves to your code and team, manageable with clean APIs

Each path can get you to an AI core — the question is how long it will keep you there. Product-first delivers quick wins but little room to grow. Platform-first scales fast until real workloads hit its limits. Custom-first offers full control but demands stronger governance from the start. What matters is knowing the risks each path carries and which ones you're ready to manage before they turn into architectural debt.

Where Strategies Crack and How to Keep Them Steady

No AI core fails on day one. It drifts when early shortcuts collide with real scale, complex policies, and budget pressure. That's why it's not about finding a risk-free path, because there isn't one. The difference lies in knowing which problems will show up first and being ready to contain them before they start costing you speed, money, or control.

PRODUCT-FIRST OPTION RISKS

The product-first path feels comfortable early on, but its biggest risk is stagnation. Once usage grows and new AI initiatives appear, the same simplicity that made it appealing starts to limit flexibility and cost control.

CORE RISKS

- **Tier pricing shocks at growth or peak.** When traffic or inference volume spikes, usage-based pricing tiers jump unpredictably, pushing costs far beyond forecast.
  How to mitigate it: secure multi-year price protections, set usage caps and automated alerts to avoid runaway billing.

- **Limited experimentation and explainability.** Most product-first tools offer minimal control over model behavior, rules, or data exposure, slowing iteration and obscuring outcomes.
  How to mitigate it: require API-level access for rules and prompts and ensure BI-grade data exports for transparency and ongoing optimization.

Managing product-first risks comes down to treating these tools as components, not black boxes, securing cost control, transparency, and enough access to adapt as you scale.

PLATFORM-FIRST OPTION RISKS

Platform-first feels efficient at the start, but dependence builds quietly. When your roadmap or performance starts moving at the platform's pace instead of yours, control slips before you notice.

CORE RISKS

- **Roadmap gaps and feature lag.** Platforms evolve at their own pace, leaving teams waiting for critical features or connectors that usually don't arrive on schedule.
  How to mitigate it: define must-have capabilities before committing, run performance tests during evaluation, and secure written roadmap influence or feature request rights.
- **Opaque cost or latency at peak.** Shared infrastructure often hides real performance behavior, making cost and response time unpredictable under high load.
  How to mitigate it: run peak-load simulations, set contractual SLOs, and include exit clauses for unmet performance guarantees.
- **Lock-in.** Deep integration into platform-specific APIs or data formats can make migration difficult and expensive.
  How to mitigate it: enforce exportable data formats, build API facades around critical workflows, and negotiate renewal flexibility from the start.

Mitigating platform-first risks means designing for independence, using the platform's speed without surrendering ownership of performance, data, or future choices.

CUSTOM-FIRST OPTION RISKS

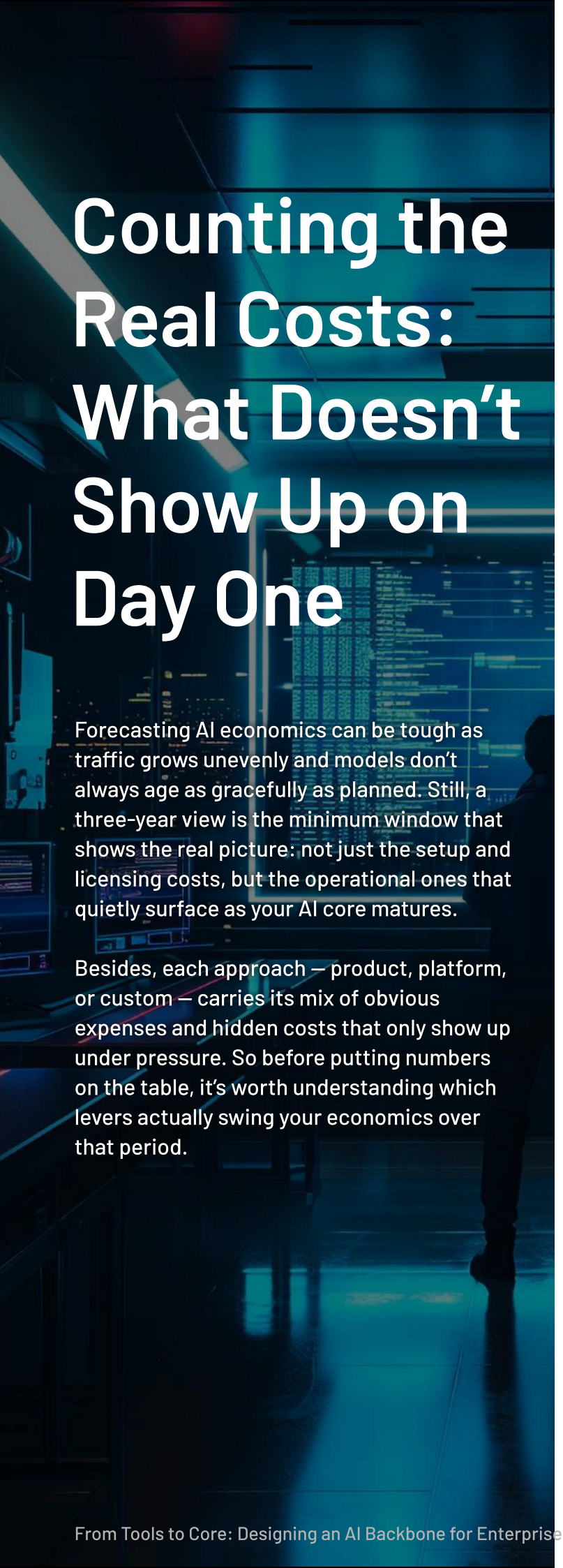
Custom-first gives you full control and full responsibility. The risk isn't in what you build, but in how consistently you can operate and evolve it. Without disciplined execution and governance, control quickly turns into overhead.

CORE RISKS

- **Execution risk and talent gaps.** Building from scratch demands a mature team and a tight delivery scope; overextending early often derails momentum.
  How to mitigate it: start with a lean core team, phase workloads gradually, and enforce hard SLO gates at each stage.
- **Operational burden.** Every layer — data, model, infrastructure — becomes your responsibility, and without strong ops readiness, stability suffers.
  How to mitigate it: create detailed runbooks, align on-call capacity to peak loads, and schedule chaos or peak drills regularly.
- **Cost drift.** As workloads grow, inference and storage costs can expand silently without structured governance.
  How to mitigate it: set token and storage budgets, apply cache tiers, and use pre-compute windows to control spend without losing performance.

Keeping a custom core sustainable comes down to two things: disciplined design and the right team behind it — the kind that turns control from a risk into your biggest advantage.

Every approach carries its own price for speed and control: you either pay upfront in governance or later in limitations. Speaking of prices...



Counting the Real Costs: What Doesn't Show Up on Day One

Forecasting AI economics can be tough as traffic grows unevenly and models don't always age as gracefully as planned. Still, a three-year view is the minimum window that shows the real picture: not just the setup and licensing costs, but the operational ones that quietly surface as your AI core matures.

Besides, each approach — product, platform, or custom — carries its mix of obvious expenses and hidden costs that only show up under pressure. So before putting numbers on the table, it's worth understanding which levers actually swing your economics over that period.

SENSITIVITY DIALS

These factors shape your AI core's cost curve because they expose how your architecture handles growth: every new market, product, or traffic surge tests where performance ends and extra spending begins.

- **Peak multiplier (BFCM):** raises per-request costs and latency risk in shared systems, while rewarding custom setups that use caching or precompute.
- **Multilocale factor:** as new markets launch, embeddings and storage scale non-linearly, and translation or compliance reviews add recurring overhead.
- **Catalog growth:** expanding product data inflates vector size, which impacts latency and compute cost. Efficient indexing and denormalization keep it in check.
- **Experimentation cadence:** faster iteration loops compound business value, but only if your chosen lane supports frequent model and rule updates.
- **Traffic growth & model pricing:** rising usage and model updates can trigger tier jumps in SaaS or force infrastructure upgrades in custom builds.

Understanding these dials sets the baseline. Now it's time to see how each approach turns those same pressures into either predictable costs or hidden surprises.

OBVIOUS VS. HIDDEN COSTS OF AI PATHS

PRODUCT-FIRST OPTION

Hidden costs in product-first setups appear when scaling, new markets, or custom needs start pushing against the limits of what the product was built to handle.

- **OBVIOUS COSTS:** subscription tier, usage and overage fee, initial integration setup.
- **HIDDEN COSTS:**

  1. Tier jumps at peak and localization surcharges: seasonal traffic or market expansion pushes you into higher pricing tiers and per-locale fees.

  2. Experimentation limits: built-in A/B testing constraints slow learning cycles, creating opportunity costs in optimization speed.

  3. Parallel tools: you'll still need external systems for feature flags, observability, and offline evaluation, adding recurring spend.

  4. Vendor bundle creep: covering niche needs often requires buying add-on modules, each with its own pricing model.

  5. Export and lineage work: additional engineering needed to extract, trace, or audit data outside the product for compliance or analysis.

PLATFORM-FIRST OPTION

The further your AI use cases drift from the platform's core templates, the more you spend translating your needs into its language through extensions, add-ons, or extra layers of process.

- **OBVIOUS COSTS:** platform licenses, per-seat or per-run usage fees, integration and setup work, support tier
- **HIDDEN COSTS:**

  1. Connector gaps: missing or unreliable connectors force "mini-builds" and custom mappings that add maintenance overhead.

  2. Premium guardrails and observability: advanced monitoring, governance, or security layers often sit behind higher pricing tiers.

  3. Data egress and storage: moving data between the platform and your analytics or BI tools incurs hidden transfer and storage fees.

  4. Roadmap workarounds: missing native features lead to parallel processes or tools that add unplanned complexity and cost.

  5. Peak-time inefficiency: shared infrastructure means you pay for burst capacity you can't fully optimize or tune.

CUSTOM-FIRST OPTION

Over time, the real expense of custom core isn't in what you build, but in keeping it observable, governed, and tuned to perform at scale.

OBVIOUS COSTS: engineering build and development effort, infrastructure and storage, inference and compute, SRE and operational support.

HIDDEN COSTS:

1. Observability: skipping early investment in monitoring leads to higher incident and debugging costs later.

2. Model lifecycle: ongoing retraining, evaluation sets, and drift monitoring introduce steady operational overhead.

3. Vector growth: as catalog size multiplies across locales, embedding storage and retrieval costs rise unless you plan compaction and retention strategies.

4. Governance processes: implementing and maintaining change control, audit, and brand safety reviews consume ongoing time and resources.

However, with hidden costs, the custom-built option brings hidden savings as well.

HIDDEN SAVINGS:

1. Reuse across use cases: one AI core supports many workloads, lowering unit costs as adoption grows.

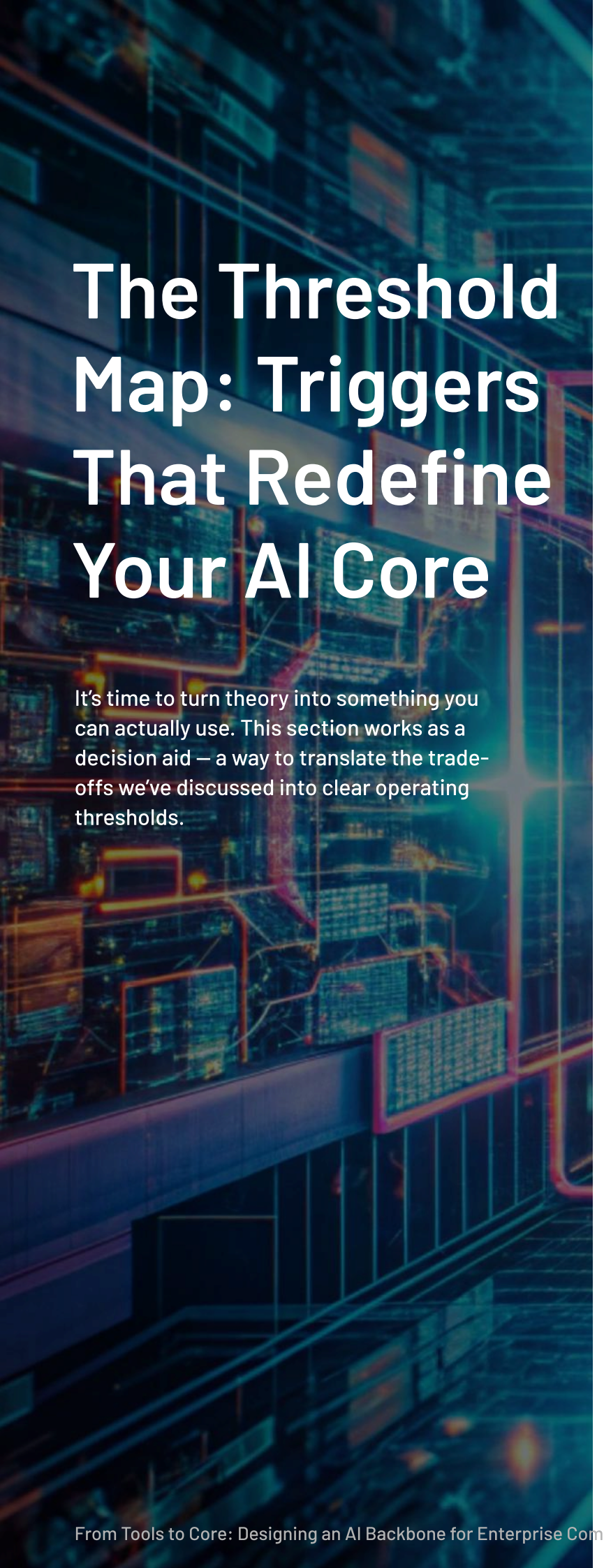
2. Avoided tier jumps and overages: owning infrastructure prevents the sudden pricing spikes common in product or platform models.

3. Performance tuning: optimized caching and precomputing reduce cost per query while keeping peak performance stable.

EXAMPLE FROM PRACTICE

A global electronics distributor asked the Expert Soft team to build a custom AI chatbot to automate lead qualification across markets. Once established, the same AI core was easily reused for other functions: onboarding, post-sale support, and supplier communication. This way, instead of building a dedicated AI solution from scratch, the client reduced costs by adapting the existing custom logic.

Even knowing every hidden cost, your AI economics stay predictable only when the entire commerce architecture is built to be predictable. Explore the ways to lower infrastructure costs without compromising system excellence.



The Threshold Map: Triggers That Redefine Your AI Core

It's time to turn theory into something you can actually use. This section works as a decision aid — a way to translate the trade-offs we've discussed into clear operating thresholds.

Each trigger below will mark a condition where the demands on your AI core shift, whether it's performance, scale, or integration depth. Under each, you'll see where Product-first, Platform-first, and Custom-first still fit those conditions and where they start to fall short.

Explore them as readiness checkpoints: if four or more triggers land in the Custom-first option, it's a clear signal your AI core should become something you own, not rent.

PEAK LATENCY & SLOS

Trigger: any AI-assisted path must hold P95 < 120 ms (autosuggest) or P95 < 200 ms at peak, not just off-peak.

- **Product-first fits when:** the vendor can prove peak SLOs in your region/tenant, and your experience is mostly read-only (little dynamic decisioning).
- **Platform-first fits when:** P95 targets are softer, or you can tolerate temporary feature throttling during peaks.
- **Custom-first fits when:** you need deterministic performance under spiky load, with control over caches, precomputed windows, and warm pools.

MARKETS & BRANDS SCOPE

Trigger: ≥ 5 locales or ≥ 3 brands with distinct policies/assortments/compliance rules.

- **Platform-first fits when:** most policies can be centralized, and minor brand differences are acceptable via config.
- **Product-first fits when:** a single brand/locale dominates traffic and others can inherit nearly the same rules.
- **Custom-first fits when:** you need a policy layer that expresses global logic along with local overrides without duplicating rules/content.

EXAMPLE FROM PRACTICE

A custom AI chatbot implemented by the Expert Soft team had to operate across markets with vastly different privacy requirements, including GDPR in Europe, regional disclosure rules in CIS, and additional consent logic in Asia. Instead of fragmenting the setup with separate configurations for each region, the system used a single policy layer that recognized user location, applied the right data-handling flow, and logged consent locally.

This approach kept one global logic intact while letting regional rules evolve independently, something that would have required multiple cloned bots or hard-coded exceptions in a product or platform setup, but came naturally with a custom core.

INTEGRATION DEPTH

Trigger: need near-real-time read/write flows, contract-tested APIs, and feature-flag control from your platform.

- **Platform-first fits when:** webhooks and polling latency are acceptable, and most flows are read-only.
- **Product-first fits when:** a few shallow integrations cover 80% and change frequency is low.
- **Custom-first fits when:** you require deep, low-latency integrations and shared core services that other systems can call.

EXAMPLE FROM PRACTICE

An AI chatbot built on a custom core integrated seamlessly with enterprise systems through modular adapters, connecting to CMS, CRM, and ERP platforms like SAP, Dynamics 365, Salesforce and Zendesk via standardized REST/gRPC interfaces. This allowed data gathered in chat to flow instantly into business systems in a clean, structured format.

Where a product or platform setup would depend on vendor connectors or batch syncs, the custom core handled integrations as native features — fast, reliable, and easy to extend to new tools without rebuilding the logic.

CATALOG & MERCHANDISING COMPLEXITY

Trigger: your catalog includes configurable products, regulated attributes, or logic that ties ranking to margin, inventory, or compatibility rules.

- **Platform-first fits when:** catalog is mostly flat, facets and simple business rules suffice.
- **Product-first fits when:** the product's native rules engine matches your needs with minimal extensions.
- **Custom-first fits when:** business logic must be first-class in the decisioning layer (blend of rules + ML) and reused across journeys.

DATA RESIDENCY & COMPLIANCE

Trigger: multi-region operations require strict data residency, retention, and per-decision audit compliance.

- **Platform-first fits when:** residency can be met by vendor regions/VPC options, and audit needs are dashboard-level.
- **Product-first fits when:** standard SaaS controls satisfy compliance, and exports are enough for audits.
- **Custom-first fits when:** you must enforce policy in your environment, keep full decision lineage, and tailor controls per market.

TRAFFIC SHAPE (BFCM AND LAUNCHES)

Trigger: peak is ≥ 5× baseline or highly bursty (campaigns, drops, TV moments).

- **Platform-first fits when:** peaks are mild, predictable, and vendor autoscaling is contractually covered.
- **Product-first fits when:** you can pre-warm capacity through the vendor and accept conservative fallbacks.
- **Custom-first fits when:** you need cost-efficient peak resilience (cache priming, precompute, rate guards) tuned to your actual pattern.

If four or more triggers land in the Custom-first column, it's a sign your business has outgrown packaged limits. Platform- and Product-first options still have their place, but tactically, while your real advantage comes from owning the core.

Custom AI Core Readiness Scorecard

If you've started to seriously consider the Custom-first path, there are still some things to think about. Even if Custom-first makes strategic sense on paper, it only pays off when the foundations beneath it are solid enough to sustain it.

This scorecard turns that question into something you can measure.

It breaks readiness into five dimensions — data, integration, operations, team, and business clarity — and helps you see where your organization stands today. Rate each from 0 to 3, then read your total as a signal of how close you are to safely owning your AI core instead of renting it.

DIMENSION	0	1	2	3	SIGNAL
Data readiness	Fragmented attributes, inconsistent events	Partial / ETL-only pipelines	Clean PIM, consistent events	Governed schemas, enrichment loops, golden attributes	At 2-3, a custom core compounds value across use cases
Integration readiness	Few APIs	Manual releases	Stable APIs, CI/CD, flags	Contract-tested, evented architecture	At 2-3, you can meet strict SLOs with deep integrations
Ops & governance	No SLOs	Basic dashboards	SLOs, alerts, security reviews	Runbooks, peak drills, cost budgets	At 2-3, running a bespoke core is safe
Team capability	No owner	Named owner, limited bandwidth	PO + Eng lead, A/B culture	Dedicated ML ownership, weekly iteration	At 2-3, iteration speed becomes your advantage
Business clarity	Broad ambitions	Directional KPIs	Quantified KPIs, baselines	KPI targets with time-to-impact SLAs	At 2-3, custom becomes economically defensible

HOW TO TRANSLATE:

- **0–6** — shore up foundations. Strengthen data, ops, or integration. If you start with Product/ Platform, design for easy migration.
- **7–11** — ready to benefit from a custom core. You can safely begin building a custom core — phase workload gradually.
- **12–15** — custom-first strongly indicated. You're ready for full ownership. Use Product modules tactically for commodity needs.

The Anatomy of a Reliable Custom AI Core

So, you've decided the Custom-first path is right for you, and your scorecard says the foundation can handle it. The next question to explore is what it actually takes to make that decision to work in practice.

ROLES

Behind every successful custom AI core is a team built for both depth and coordination.

You need a **product owner** who connects business goals to delivery rhythm, and a solution **architect** who ensures the system fits seamlessly into your landscape.

Around them, **engineers:** data engineer, ML/ranking Engineer, LLM engineer (RAG/prompting), back-end (APIs), front-end, QA automation. DevOps/SRE and security specialists are also required to make reliability a habit, not an afterthought.

REALISTIC DELIVERY BANDS

The pace of delivery matters as much as the design itself. In practice, laying the foundation takes about 4–8 weeks, just enough to make the core usable.

Within 8–12 weeks, you should see the first value from one or two live journeys.

After that, scaling happens quarterly, adding new journeys or markets onto the same backbone.

REUSABLE COMPONENTS

A custom AI core isn't built from scratch every time. It's assembled from reusable planes that keep data, logic, and operations consistent across use cases. Together, they form the foundation for scale without chaos.

- **Data plane** handles ingestion from SAP Commerce, PIM, or ERP, enrichment, and validation.
- **Model plane** runs ranking and recommendation services with LLM adapters and guardrails.
- **Orchestration plane** manages APIs, routing, and experimentation via feature flags and interleaving.
- **Ops plane** ensures observability, cost governance, and SLO discipline with clear runbooks.

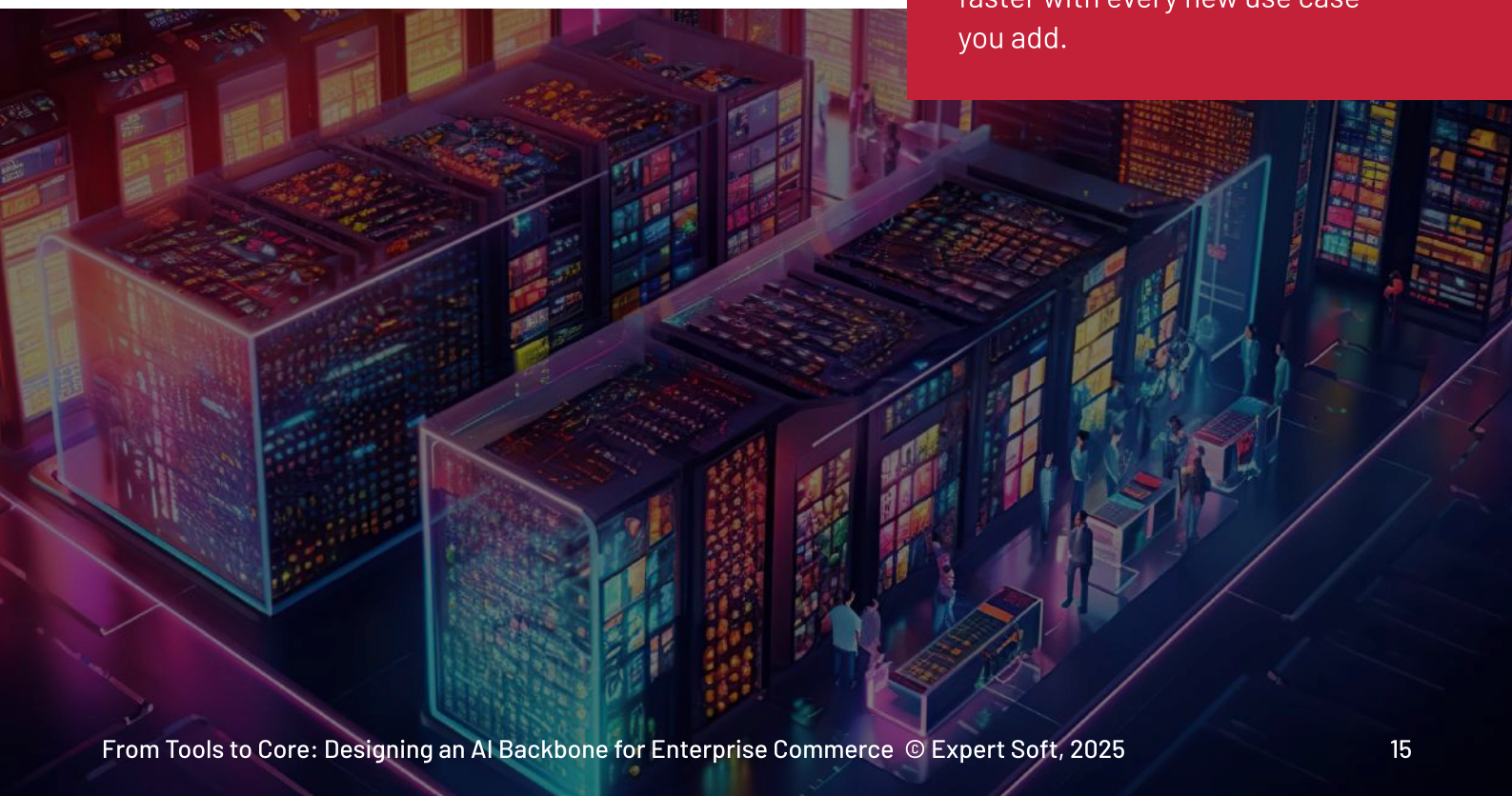
NON-FUNCTIONALS THAT MATTER IN ECOMMERCE

In ecommerce, speed is the difference between a cart and an abandoned session. That's why a custom AI core must hit tight P95 and P99 targets across every interaction.

But speed alone isn't enough. When your system operates across regions, multi-locale policies keep responses compliant and relevant. And when traffic spikes, peak readiness — through caching and precomputing — keeps performance predictable.

Finally, audit trails ensure every decision can be traced, proving your AI runs fast, fair, and accountable.

Once those pieces are in place, the real advantage shows: a system that grows stronger and faster with every new use case you add.



AI Core Game Plan

After all the trade-offs, risks, and economics, what matters most is clarity — seeing on one page how each path translates into real actions, measurable checkpoints, and signals that you're moving in the right direction.

	PRODUCT-FIRST (SAAS/OFF-THE-SHELF)	PLATFORM-FIRST (AI "STUDIO" / PLATFORM)	CUSTOM-FIRST (BESPOKE AI CORE)
When It Shines	Win fast on standardized problems; minimal integration; acceptable with relaxed SLOs, few markets, and limited experimentation needs.	Rapid internal tool creation with governance; good when you can live within platform primitives, have moderate scale, and want a single place to orchestrate multiple lightweight apps.	When you need tight peak SLOs, multi-brand/locale policy, deep integrations, weekly iteration, strict residency/audit, and when you expect many use cases over 24-36 months.
Commitment Plan	contract peak SLOs, export/ portability rights, usage caps	governance (roles, audit), PIM/ERP/ Commerce connection, data residency	fix SLO targets (P95/P99), lock data contracts, staff lean core team
	stand up feature flags and basic observability outside the product for future flexibility	ship 1–2 internal apps; implement performance testing at the projected peak and set fallback policies	stand up the core skeleton (data plane, policy layer, ranking/LLM adapters, API facade, observability)
	prove KPI lift vs. baseline; document a migration path to core if triggers fire	close gaps (connectors, observability, AB testing). Record which workloads would move to a core if the limits block progress	cut over 1–2 high-impact journeys onto the core; run peak drills and rollback tests; set cost budgets/ alerts

FINAL CHECK

Use this checklist to confirm that your chosen path, whether product, platform, or custom, is truly ready to scale without surprises.

- [ ] KPI targets and time-to-impact are defined
- [ ] Peak P95/P99 requirements are written and testable
- [ ] Weekly changes can ship safely (flags, canaries, rollback)
- [ ] Data contracts and event pipelines are under control
- [ ] AI decisions are observable and auditable
- [ ] Cost governance is active (budgets, alerts, peak drills)
- [ ] Use cases are labeled "differentiating" vs. "commodity"
- [ ] Ownership model and partner support are explicitly accepted

With the right foundations and the discipline to follow them, any path can deliver lasting value. And with a reliable partner beside you, that path simply gets smoother and faster.

About Expert Soft

Expert Soft is a targeted ecommerce software delivery company, partnering with Fortune 500 companies and global corporations across the US and EU. With SAP Commerce Cloud and Java as our backbone, we know how to ensure scalable and high-performing solutions that can handle 1 mln requests per second, delivering a smooth customer experience.

Developing a payment engine that saved our client about $100 million in operational expenses, ensuring multi-country platform support, adapting solutions for new market entry with tailored enhancements — these are just a few of the challenges our specialists tackle.

We aim to deliver more than a software system. We aim to deliver tailored solutions that maximize profitability within available resources. Our success is driven by:

TEAM STRENGTHS

- **All our engineers have a university background**
- **Specialists excel their skills in our training LABs**
- **Perfect English skills**
- **Ready to help 24/7**

CLIENTS

We work with corporations around the world with revenue of over $20 billion and 150K+ employees.

APPROVALS BY AUDITS

Our ongoing work with corporations is consistently validated through rigorous audits, both by internal teams and Big 4 consulting firms.

HIGH-LEVEL SECURITY

Approved by assessments from global companies, who are leaders in their respective industries.

BUDGET EFFICIENCY

By carefully aligning technology investments with your business goals, we ensure optimal value and cost-effectiveness.

PROFESSIONAL TEAM

No offshore outsourcing and our team's average tenure of 4+ years means you get seasoned problem-solvers, not just coders.

EXPERT SOFT EXCELS IN

- PAYMENT ENGINE
- MICROSERVICES ARCHITECTURE
- CONTENT MANAGEMENT
- REDESIGN
- E-COMMERCE PLATFORM
- HEADLESS COMMERCE
- MICRO FRONTENDS
- MIGRATION&INTEGRATION

OUR TECH CORE

FRONT-END
HTML, CSS, JavaScript (Angular, React, Vue, Next, TypeScript, Jquery), Spartacus

BACK-END
Java EE, Spring, SAP Commerce (Cloud), Node.JS.

DEVOPS
Docker, Kubernetes, CI/CD

UX/UI DESIGN
UX Research, UI Design, Figma, Adobe, Sketch

QUALITY ASSURANCE
Manual Testing, Test Automation

TARGETED DOMAINS

SHARED PATHS, LASTING ECOM VICTORIES

LET'S TALK SOLUTIONS!

EKATERINA LAPCHANKA
Chief Operating Officer
kate.lapsenco@expert-soft.com

+1 585 4997879
+371 25 893 015

PAVEL TSARYKAU
CEO & Founder
of Expert Soft

Let's connect

expert-soft.com

LinkedIn