



PLAYBOOK

Commerce Migration

A playbook for the person who has to defend the number

Move less. Decide early what dies. Treat data and search traffic as revenue work, and judge the whole thing at day 90.

CommerceStudio.ai

2026 Edition

Most replatforming projects get sold as a technical upgrade. Then month five arrives, the ERP integration is three weeks behind, and someone in finance asks why order history didn't come across.

The platform is almost never the reason it went sideways. What kills migrations is the decision to rebuild what you already had on something new, because nobody was willing to say out loud that half of it was never used.

Legacy parity is the most expensive phrase in commerce. It sounds responsible. It means you pay twice: once to move complexity you didn't need, and again in two years when you're still carrying it.

This playbook argues the opposite. Move less. Decide early what dies. Treat data and search traffic as revenue work instead of tickets at the back of the plan. And judge yourself at day 90, because launch day tells you nothing except that the site loads.

If you want a feature comparison, this isn't it.

Five ways these go wrong

1. You migrated the old site instead of the business

The team pulls up the legacy platform and starts cataloguing. Every feature, every workflow, every one-off somebody built in 2019 for a customer who's since churned. It all goes on the list, because taking something off the list means making a decision, and decisions need an owner.

You land on a modern platform that behaves exactly like the old one, four months late, and nobody can say what got better.

The fix is a usage audit, and it's uncomfortable. Pull the real numbers on what gets touched. You'll find features nobody has opened in a year. Kill them during scoping, because once you're in delivery the deadline pressure will make you keep everything.

2. You assumed the data was fine

It isn't. It's never been tested.

A medical device manufacturer I worked with had over 5,000 SKUs and no real data governance behind them. No attribute model worth the name. On the legacy platform that was invisible, because nobody was asking the catalog to do anything clever. The moment you want faceted search, product grouping that makes sense to a buyer, or a filter that returns the right twelve items instead of four hundred, the absence of an attribute model stops being a data problem and starts being a merchandising problem.

Moving unstructured product data to a better platform gives you unstructured product data on a better platform.

Audit before you scope. Not during. The teams that get burned here are the ones who found out what was in the catalog at the same time they were trying to load it.

3. Search traffic showed up in week 40

URL structures, redirects, metadata, structured data. These are revenue assets, and they get handled by whoever has capacity in the final sprint.

Organic traffic drops after a badly handled migration, and it takes months to come back. You'll be explaining that drop to people who were told the new site would be faster and better.

Give it an owner with a name, and give them the URL map before the build starts, not after. And in 2026 there's a second reason to care: if your product data isn't structured for machine consumption, AI answer engines can't read you either. You can lose that visibility in the same week you lose the rankings, and it won't show up in the same report.

4. You believed the integrations would transfer

They won't. API structures differ. Authentication models differ. Sync timing that was fine on the old platform breaks on the new one because the rate limits are different.

The pattern is always the same: the integration inventory gets built from documentation that's two years stale, the pre-built connector is assumed to cover the requirement, and the gap turns up in production where fixing it costs the most.

Re-validate every touchpoint against the new platform's actual architecture. Every one. Including the ones somebody swears are simple.

5. You trained the team after go-live

Commerce platforms change how your people work, not just how customers buy. Merchandising, content, order operations, reporting. All of it shifts.

When the first time your operations team sees the new platform is launch day, adoption stalls and they build spreadsheets to route around it. Those spreadsheets become permanent. Six months later you're paying for a platform your team is working against.

The seven dimensions

The framework exists so decisions get made in the right order, with the weight sitting where the risk is. These weights are a starting point. Tune them to your situation rather than accepting them.

Dimension	Weight
Business outcome clarity	20%
Scope and legacy rationalization	15%
Data migration strategy	20%
Integration re-architecture	15%
SEO and traffic preservation	10%
Cutover planning and risk	10%
Post-migration optimization	10%

Business outcome clarity. What does this migration let the business do that it can't do now? Vague answers send teams straight back to legacy parity, which is how you end up at failure one. The test I use is blunt: two people from different departments, asked separately, should state the goal the same way. When they can't, you don't have a goal. You have a budget.

Scope and legacy rationalization. Not everything deserves to move. Sort the catalog of features into migrate, rebuild, and retire, and make someone sign the retire list. The argument you're avoiding now is the argument you'll have in month four with less time.

Data migration strategy. Product, customer, order history, pricing, content, taxonomy. Each needs a quality assessment, transformation rules, and a dry run with validation. Decide what genuinely has to come across and what can be archived. Order history is where this gets political, and it's worth having that fight early.

Integration re-architecture. Full inventory, validated against the new platform rather than the old documentation. Which connectors are real, which need building, what happens when one fails at 2am.

SEO and traffic preservation. Complete URL mapping, 301s tested before launch, metadata and structured data carried over, and a performance baseline recorded while the old site is still up. You can't prove you protected traffic if you never measured it.

Cutover planning. DNS, CDN, certificates, payment processor. The rollback plan matters more than the runbook, because the runbook assumes things go to plan. And don't schedule this into a peak period, however good the pressure to launch before the quarter closes feels.

Post-migration optimization. What happens in the 90 days after. Covered below, because this is the part that decides whether anyone calls the project a success.

Four shapes a migration takes

Most migrations look like one of these. Filing yourself neatly into a category matters less than knowing which risks are pointed at you.

Monolith to modern SaaS. Magento, Hybris, ATG, or something heavily customized, moving to Shopify Plus, BigCommerce, or similar. Your problem is years of accumulated customization that made sense when it was built. Be aggressive about retiring it, phase the cutover, and give data migration its own workstream with a real owner.

Platform to platform. Modern SaaS to modern SaaS, usually driven by a specific capability gap or B2B requirements the current platform can't handle. Lower debt, but the app ecosystem differences will surprise you and checkout deserves disproportionate testing. Keep the scope pinned to the gap that triggered the move.

B2C adding B2B. An existing consumer operation bolting on trade selling. Contract pricing, account hierarchies, approval workflows, purchase orders. Decide whether your current platform genuinely supports B2B or just claims to, and get ERP integration and the pricing engine moving first. Those are the long poles and everything else waits on them.

Headless or composable. Usually experience-led, and usually chosen by someone who's seen a great demo. The honest question is whether you have the engineering capacity to operate it, not build it. Operating responsibility moves from the vendor to you permanently. Treat checkout as the highest-risk piece in the architecture and never assume it's solved.

Cutover strategy

There's no correct answer here. The worst outcome is defaulting to big bang because nobody raised the alternative.

Strategy	Best for	Strengths	Tradeoffs	Watch for
Big bang	Simple migrations, low integration complexity, strong test coverage	Clean break, single date, no parallel running costs	All risk on one day, no incremental validation	Cutover during peak, no rollback plan
Phased	Multiple markets, brands, or product lines	Risk reduced per phase, learning between phases	Longer overall, dual-platform operating costs	Phases that never finish
Strangler fig	Enterprise modernization where disruption is unacceptable	Lowest risk per increment, continuous delivery	Highest complexity, needs a solid routing layer	Stalling halfway and running two platforms forever
Parallel running	High-revenue operations where downtime isn't an option	Zero-downtime, validated against live traffic	Heavy operational overhead, data sync complexity	Parallel running quietly becoming permanent
MVP and iterate	Speed to value over completeness	Fastest to market, validated learning	Incomplete at launch, needs stakeholder discipline	MVP scope growing until it isn't minimal

The first 90 days

A migration is successful if the business is performing better 90 days after launch than it was 90 days before. That's the only measure that survives contact with a board.

Days 1 to 14 are for stabilization. Watch production, fix what broke, confirm the data landed correctly, verify the critical integrations under real load. Ship nothing new. The goal is confidence, and every new feature in this window buys you a fresh unknown.

Days 15 to 45 are for optimization. Conversion, site speed, search tuning, the obvious UX fixes the team spotted in week one. This is where you start showing movement against the pre-migration baseline, which is why you needed that baseline.

Days 46 to 90 are for acceleration. Phase-two capability, deferred integrations, the things you consciously cut during scoping. A team that isn't moving faster by day 90 than it was before the migration tells you something in the framework got skipped.

Stop and re-plan if

- Legacy parity is the stated goal
- Nobody has audited the data
- There's no URL map

- There's no rollback plan
- The integration architecture is assumed to transfer
- The team hasn't touched the new platform
- Nobody can name the success metrics for day 90

Any one of these is recoverable. Three together means the plan isn't real yet.

Where we come in

We assess current state against this framework, quantify what the migration will actually cost in effort and risk, and give you a sequenced plan. Then we help run the workstreams that decide the outcome: data migration, integration, SEO, cutover.

We've lived through every pattern in this document, including from the wrong side. That's mostly what you're buying.

If you're planning a migration, or cleaning up after one that didn't go the way it was sold, that's a conversation worth having.

Ready to start?

info@commercestudio.ai

No pitch. Just a conversation about where you are and where you're trying to get to.

