Speed is the new normal

By Dr Brian Russell

If you're building a product in 2026 and your plan still assumes "we'll ship in December," you're already behind. Not because everyone suddenly became smarter. Because **expertise is being commoditised** and this shifts implementation paradigms. An expert engineer with deep domain knowledge *may* now be slower than a junior builder with strong AI tooling and an obsession with shipping. Even more interesting: domain experts who can now "vibe code" are in a new position entirely. They don't need to spend months translating reality into requirements for someone else to interpret. They can build the artifact directly, watch users interact with it, and iterate in tight loops.

This isn't only about software products. It's about the *whole business surface area*:

- the website and landing pages
- lead magnets, onboarding flows, email sequences
- sales collateral, demos, investor updates
- internal tools, dashboards, and reporting
- automations across CRM, support, billing, and operations

The companies that win are increasingly the ones that can **ship the business** as fast as they ship the code.

Deep computational algorithms that used to take weeks of learning and implementation can now be coded in minutes, refined in hours, and tested the same

day. Not always perfectly. Not always elegantly. But fast enough to create a feedback loop that drives the next build.

And that's the point.

The cost and friction to build, ship, and sell has collapsed for a certain class of products—especially products that can reach users directly, collect continuous feedback, and scale with software instead of headcount. If you're still running 2024 playbooks for one of these products, you'll lose to teams that aren't.

The thesis

In 2026, speed is not a nice-to-have. It's a strategy.

Speed doesn't mean rushing. It means designing your product, team, and economics so you can learn faster than competitors and turn learning into shipped artifacts—software, content, workflows, and sales assets. The winner is often not the team with the best initial idea. It's the team with the fastest learning loop that compounds into a better product *and* a sharper go-to-market.

Why this is happening

A lot of what used to be "the work" has turned into commodity output.

Scaffolding, glue code, setup, wiring, boilerplate UI, database wrappers, deployment scripts, testing harnesses—these were the invisible tax that slowed everyone down. The same is now true for business outputs: first-pass landing page copy, pricing pages, FAQs, sales decks, demo scripts, nurture sequences, even basic customer support playbooks. Tools can create decent first drafts quickly, which means humans can spend more time on the high-leverage parts: choosing the problem, defining the user, shaping the workflow, designing the experience, deciding what to measure, interpreting the signal.

That changes who wins.

A traditional team might spend months polishing a spec, aligning stakeholders, writing tickets, and sequencing sprints. Meanwhile a smaller, AI-augmented team builds something imperfect but real, ships it, and starts collecting data. They don't win because they're "better at coding." They win because they're better at discovering truth while the slower team is still debating what truth might be.

The uncomfortable truth

If you could have shipped faster, and you chose not to, the market will punish you for it. Not emotionally. Not morally. Just mechanically.

Because the teams shipping faster are learning faster, and learning is the compounding advantage.

The 4-speed framework

When I look at teams that move fast and stay sane, they usually win on four dimensions. Miss one and you can still ship quickly for a while, but you'll struggle to turn speed into a business.

1) Product speed: ship to learn

Your first job isn't to be right. It's to get the feedback loop running.

Requirements are no longer something you "finish." They're something you discover—by putting a real artifact in front of real users. That includes software, but it also includes marketing and business artifacts: a landing page that forces you to explain the value, a pricing page that forces you to pick a business model, an onboarding flow that forces you to decide what "success" means for a new user.

A practical test: can you get a real user interaction within two weeks? Not a demo to your friends. Not internal "we think it's cool." A real user doing a real workflow, clicking a real page, replying to a real offer, or paying for something.

If you can't, you don't necessarily have a bad product. But you do have a speed problem. And speed problems compound.

2) Team speed: the builder/PM merge

Developers and product managers are converging. Marketers and product are converging too.

Modern teams need deep product thinking and full-stack execution. The good news is that many mechanical parts can be augmented by AI: not just code, but copy, designs, scripts, analytics instrumentation, and internal automation. "AI-native" doesn't mean replacing people. It means treating tools like Claude/Codex-style assistants as force multipliers, so a small team behaves like a much larger one.

If you can spend a few hundred dollars a month on tools that make you meaningfully faster, that's often cheaper than hiring early—especially before you've found product–market fit. Hiring is a long-term bet. Tools are a short-cycle multiplier.

A cultural shift follows: the fastest teams are increasingly the ones where domain experts can build directly, not just specify. The moment you remove the translation layer between "knowing the problem" and "building the artifact," speed and quality both improve.

3) Go-to-market speed: design for shorter paths to revenue

Enterprise sales cycles aren't magically shrinking. Many B2B motions still look like 12–18 months, with procurement, security reviews, stakeholder alignment, and budget approvals.

So ask a hard question early: can the product start as B2C (or prosumer) and later expand into B2B? Call it B2C2B if you like. The label doesn't matter. The mechanics do.

That shift changes everything. You can market while you build. You can iterate pricing early. You can earn revenue before procurement. You can build your brand and distribution while the product is still forming. You can let individuals pull it into organizations once it's already proving value.

This doesn't remove governance. But it can drastically change the pace of learning and adoption.

4) Economic speed: token costs, pricing, and the "free" trap

AI changes unit economics in a way many teams underestimate.

Two things matter immediately.

First, token costs are cost of goods sold (COGS). If each workflow costs you money to run, you need to know what it costs, where it spikes, and how it scales with usage. This applies not only to the product, but also to business tooling: customer support automation, prospecting workflows, content generation, reporting pipelines.

Second, free can be either a growth engine or a cash leak. Freemium is not a philosophy. It's a design choice. If the product's value is high but costs are variable, you need pricing that can breathe with usage. Limits, tiers, credits, or pay-per-use mechanics become part of product design.

Speed isn't just shipping fast. It's shipping fast without creating a business you can't afford to run.

What's different from 2024

The breadth of what's possible has exploded. AI tools are now good enough to be used in production. Platforms for building and shipping web and app products have reduced the level of expertise needed to get something live. If you have domain expertise and a product idea, you can ship quickly.

Someone can take a day, learn a modern deployment path, and ship something real. Tools like Replit (and similar platforms) can make prototyping almost unfairly fast—not just for product code, but for marketing pages, signup flows, onboarding, and internal dashboards.

But there's a trade. Fast platforms accelerate early speed. Their leverage fades as complexity grows.

So the move is: use fast defaults to get learning velocity—then deliberately invest in architecture when the product starts proving itself. Prototype quickly, then migrate to a production-grade stack when the signal is real.

Examples, so this isn't abstract

A lot of speed wins don't come from a magical feature. They come from designing the system so distribution and feedback are built in.

A "viral loop" product intentionally creates sharable outputs: branded reports, dashboards, collaboration artifacts, or before/after results that people naturally forward. Now your users are doing marketing while you're doing product.

A B2B product with a B2C entry gives an individual a clear win—time saved, clarity gained, a reusable asset created—and lets that person pull it into the organization. That changes the sales motion because you arrive at governance with evidence, not theory.

What I'd do this week

If you want speed in 2026, don't start with a roadmap. Start with constraints.

Pick one problem you need solved and prototype it fast. Build the smallest workflow that produces a user-visible outcome. Put a hard ceiling on MVP scope. Instrument the product so you can see what users do, where they stop, and what they ask for. Then apply the same approach to the business wrappers: landing page, pricing, onboarding, and the first sales motion.

A few practical defaults:

- Prototype one workflow end-to-end, not ten half-workflows.
- Ship a simple website that makes the value legible, not "perfect branding later."
- Build sharing/collaboration/report outputs early if virality matters.
- Track token costs per key action and decide what to cache, limit, or price.
- Build feedback into the product and the business: "request feature," "report issue," plus lightweight admin analytics.
- Treat deployment as part of development. If it's not live, it's not real.

And ask a better question than "Can we ship by December?"

Ask: "How do we ship an MVP by March?"
Not because March is magic, but because a short deadline forces you to build what matters and drop what doesn't.

Caution: it's still business

Speed changes what's possible, but it doesn't repeal reality.

You still need customers. You still need trust. Human relationships still move at human speeds. Procurement still takes time. Some domains still have long validation cycles. Coordinating and planning with a team still requires communication, alignment, and judgment.

Some parts of the business can be 10× faster now. But the market will still teach you the same lessons it always has. The difference is you can reach those lessons earlier, and adapt before the window closes.

Final thought

If you're not going fast enough, you will lose to lesser products with faster iteration, because they will learn earlier and adapt sooner. Speed is not a vanity metric. It's a learning advantage that turns into product advantage and then into market advantage.

Build virality into the product if you can. Build market feedback into the product even if you think you don't need it. And treat shipping as the normal state of the company—software, website, marketing assets, and internal tools all moving together.

If you're building an AI-enabled product and want a fast sanity check on interaction design, unit economics, and how to ship without losing trust, email me: