Products people pay for the second time.

By Rahul Jindal

Most product teams optimize for the moment of conversion. The signup, the demo, the first purchase. The metric the team is held to runs through the moment of acquisition, and the demo gets better and better at producing it. The rest of the product flatlines.

The honest test of a product is not whether the user signs up. It is whether they come back, pay again, and tell a friend. The first sale is funded by hype, by curiosity, by your marketing budget. The second sale is the product itself, with no makeup on. Most products do not pass.

This is the practice I keep coming back to: building products that pass the second-sale test. The version of the product that closes the round and the version that retains are different products. Founders who do not separate them ship the round version and lose to the field version.

> ***"The first sale is funded by hype. The second is the product itself, with no makeup on."***

Why most products fail the test

Three patterns, in roughly the order I encounter them in the field.

1. **The job is wrong.** Clayton Christensen's framing: people don't buy products, they hire products to do a job. Most products hire themselves to a job that sounds plausible in the kickoff and turns out to be wrong in the field. The user signed up because the marketing made the job feel close enough; they did not come back because the actual job was different. Christensen's ur-example is the milkshake hired to make morning commutes less boring; the marketing thought it was sold to kids.
2. **The product gets worse as the user gets better.** Most products are designed for the new user. Empty-state UI, onboarding flow, tooltips, help text, all tuned for week

one. By month three, the user has internalized the product's mental model and the same UI feels patronizing. The expert UX never gets built. Power users churn quietly; the product flatlines into novice-only territory; the team mistakes the resulting demographic for the natural ceiling.

3. **The product flatlines as the user's situation moves on.** Even when the job is right and the expert UX exists, the product can flatline if the user's situation changes and the product does not. Notion for a 5-person team is great; Notion for a 50-person team needs different defaults the product never ships. Mailchimp for a 1,000-person list is great; Mailchimp for a 1M-person list needs different infrastructure the user discovers too late. The product gets left behind, and the user finds something else.

The four engines of compounding

Products that pass the second-sale test almost always have at least one of four engines running underneath. The naming is mine; the underlying ideas are scattered across Christensen, Hamilton Helmer (7 Powers), Marty Cagan, Teresa Torres, and a generation of internet observers.

1. **Skill compounding.** The product gets better as the user gets better at it. The user's investment in learning the product becomes a switching cost. Photoshop, Excel, Vim, Figma. The expert UX is real and shipped. The novice complains; the master will not leave.
2. **Data compounding.** The product gets better as the user generates more data inside it. Personalization is real and felt. Spotify, Strava, Duolingo, Notion at scale. The product on day 100 is recognizably different from the product on day 1, and the difference is the user's own behavior staring back at them.
3. **Network compounding.** The product gets better as more people the user knows are on it. The classic two-sided dynamic. Slack, LinkedIn, WhatsApp, Discord. The product on day 100 is unrecognizably different because the universe inside it has filled in.
4. **Identity compounding.** The product gets better as the user's identity around it deepens. The product itself may not change much; the user's relationship to it does. Peloton, Apple, Patagonia, RuneScape. The user becomes the kind of person who uses this product, and the leaving cost is psychological, not functional.

The teams I work best with name which of the four engines they are betting on, build the engine deliberately, and measure whether it is firing. The teams that flatline either bet on no engine, or bet on an engine they cannot actually build. Two-sided networks are the most-cited and least-buildable; identity is the most-undervalued and most-defensible.

> ***"The teams that flatline either bet on no engine, or bet on an engine they cannot actually build."***

Specific moves

1. **Define the second-sale test before launch.** For SaaS: month-3 retention plus expansion revenue. For consumer subscription: 90-day repeat. For B2B contract: NPS at month 6 plus renewal probability. The team rallies around a number that runs through the engine, not the conversion funnel. If the team cannot name the test, they have not yet decided what kind of product they are building.
2. **Run the discovery cadence as a metabolism.** Marty Cagan and Teresa Torres argued for continuous discovery; most teams treat it as a quarterly research project. The compounding teams treat it as a weekly metabolism. Talk to users every week, not because the roadmap demands it, but because the rhythm is the work. The teams that do this ship a different product than the teams that do not, and the difference is visible at month six.
3. **Find and build for the expert user.** Watch your top 5% by depth-of-use. Their workflow is the product you should be building. The novice will graduate; the expert will not regress. Building for the median user produces a product the expert outgrows and the novice never grows into.
4. **Pre-mortem the second sale.** Gary Klein's pre-mortem applied to retention: imagine the user does not come back. Walk it backwards. What was the most likely reason? What would you have seen at month one if you had been looking? The team that asks the question early ships a different product than the team that asks it after churn lands.

5. **Decompose usage from value.** A user who logs in every day might be checking, not using. Time-on-product is a vanity metric until you decompose what the user is actually doing inside it. The teams that conflate usage with value get a falsely optimistic dashboard and a real churn problem.

Failure modes

- **Optimizing for the deck, not the field.** The product that closes the round and the product that retains are different products. Both have to ship; the team that ships only the first wins the round and loses the renewal.
- **Trusting the survey over the behavior.** Users will say they love the product. Their behavior will say otherwise. Surveys feed the team's optimism; logs feed the team's clarity. Both have a role; only one tells the truth at scale.
- **Mistaking the demographic for the ceiling.** If only novices use the product, it is not because the market is novices. It is because the expert UX was never built and the experts left.
- **Building for the engine you wish you had.** Two-sided network effects sound great in the deck and almost never get built. Identity compounding is harder to articulate and easier to ship. Bet on the engine you can actually build, not the one that closes the round.

Open questions

- What does the second-sale test look like for products with annual contracts where the buyer is not the user? The buyer signed once; the users either adopted or did not. Renewal is the second sale, but the buying-vs-using gap distorts every signal in between. This is the standard B2B problem and the framework above does not yet treat it cleanly.
- How do compounding engines interact when a product has more than one running? Spotify has data plus identity. LinkedIn has network plus identity. Photoshop has skill plus a thinning network. The interaction effects are real and under-studied.

- Is there a fifth engine I am missing? Probably. Pattern-matching from twenty years finds four; literature and the next decade of products may add more. AI-native products in particular look like they may be running a new engine I do not yet have a name for.

The connection

The second-sale test is the product version of the through-line that runs through everything I work on. The work that compounds is the work whose value does not depend on the conditions of its first deployment. The product equivalent of an Adaptive Org is a product that absorbs change in its user's situation without losing the user. Every framework on this site is asking the same question at a different altitude.

Related on this site

- Why Your Product Roadmap Looks Healthy but Users Keep Churning →
- The Decay Tax: every product starts dying the day it launches →
- The bad-year test: revenue that holds when a channel breaks →