The Not-Invented-Here Tax

Idea In Short

Decide each build with one test: would a customer choose your product because of this component, or notice it only if it broke? Build the first kind in house and buy the second. Compare the vendor's license against three years of maintenance, including the senior engineer hours that never appear in a sprint estimate, and have someone outside the proposing team review that comparison. Praise a well-chosen integration as loudly as a clever internal tool, because credit shapes what engineers volunteer to build. Teams that skip this comparison pay a hidden tax in upkeep and in roadmap items that slip while senior engineers patch tools no customer sees.

Most build-versus-buy decisions get settled by comparing two numbers that don't measure the same thing. One is a vendor's annual license fee. The other is an engineer's estimate for version one of the in-house alternative, usually quoted in sprints and usually given before anyone has asked what version three will need.

Version one is rarely where the money goes. The bill arrives later, and mostly in hours, when a senior engineer spends every other Thursday patching a tool no customer will ever see and a roadmap item slips a quarter because that engineer was busy.

Not-invented-here [NIH] syndrome, the habit of discounting code and ideas that came from outside the group, has been studied in research and development [R&D] labs since the early 1980s.1 It hits capable teams hardest. A team that can build almost anything will say yes when someone asks whether it could build the next tool on the list.

Whether that team should still be maintaining that tool three years from now is a different question, and it rarely gets asked out loud.

Why the in-house version always looks cheaper

Part of the pull toward building is emotional, and it's been measured. Michael Norton, Daniel Mochon and Dan Ariely had people assemble flat-pack boxes, fold origami and build Lego sets, then asked what they'd pay for the results. Participants valued their own work above identical items made by someone else, even when theirs came out worse, a bias now known as the IKEA effect. An engineer who prototyped a component over a long weekend is no exception.

The estimate does the rest. A build estimate prices the work someone can picture today, which is the first release, while a vendor quote prices a finished product that already covers cases the buyer hasn't hit yet. The first customer to upload a 600-page scanned contract with no text layer usually finds one of them. Set a two-sprint estimate next to a five-figure annual license in a planning doc and the build wins nearly every time, mostly because its column is missing half the rows.

Document features show how wide that gap gets. Rendering pages in a browser is close to a solved problem, since Mozilla's free, Apache-licensed PDF.js has handled it inside Firefox for more than a decade.2 Teams scoping a PDF viewer for React tend to price that rendering step because it's what the demo shows. The comment threads and signature fields that legal asks for in month two rarely make the estimate. Comments need user accounts and storage behind them, and a signature the legal team will accept needs an audit trail, so each request turns into a small project with its own bug queue.

Where the bill actually arrives

A slower cost builds up the longer a team stays together, and researchers measured it decades before anyone wrote a line of React. Ralph Katz and Thomas Allen, two researchers at Massachusetts Institute of Technology [MIT]'s Sloan School, defined the syndrome as a stable group coming to believe it holds a monopoly on knowledge in its field. To test the idea, they tracked 50 R&D project groups at a large laboratory and compared each group's performance with how long its members had worked together. Performance rose for about the first year and a half, held steady for a while, and had dropped noticeably by year five. The drop tracked a fall in how often members talked to each other and to outside sources, as groups grew increasingly insular and communicated less with key information sources outside the group.

Software teams show the same drift through components only one person understands. The

engineer who wrote the in-house viewer or job scheduler moves to another team, and the code turns into the thing everyone routes around. Say a browser update breaks it. The fix runs from Tuesday to Friday because nobody remembers why the caching works the way it does, and none of those days were in the original estimate.

Opportunity cost is the biggest line on that bill, and almost no team tracks it. If a senior engineer costs $180,000 a year fully loaded, a quarter of that person's time on upkeep comes to $45,000. That can easily top the license someone turned down as too expensive. Those hours also come out of the part of the product customers actually compare against competitors, and senior attention is the scarcest thing most software companies have.

A sharper test than cost

Some rebuilds are the right call, and the best-known defense of NIH came from a software veteran. Joel Spolsky, a former Excel program manager who later co-founded Stack Overflow, argued in a 2001 essay that a core business function should stay in-house "no matter what." His example was Microsoft's Excel team, which kept its own C compiler because its motto was to find dependencies and eliminate them. His dividing line was practical, since he treated peripheral work such as the cafeteria as fair game for outsourcing. A pharmaceutical company should write its own drug-research software and buy its accounting package.3

That gives teams a question they can settle in one meeting. Would a customer choose the product because of this component, or would they only notice it if it broke? Picture a 40-person company selling claims software to insurers. Carriers sign because its rules engine settles simple claims in minutes, so that engine deserves every hour of in-house work. The viewer adjusters use to read police reports belongs in the second group, even though a broken one would stop their whole day.

Things already built are the hardest to judge. Eight months into an in-house tool, switching feels like admitting the eight months were wasted, and that's sunk-cost reasoning presented as loyalty to the codebase. The months are gone either way. The only live comparison is what the next twelve will cost under each option.

What "proudly found elsewhere" looks like

Companies that get past the syndrome usually change incentives instead of lecturing people about bias. A former managing director of SC Johnson's German subsidiary, writing in MIT Sloan Management Review, described how an air freshener the U.S. parent overlooked became a hit in Germany. An Asian subsidiary had launched it first. The German team picked it up with minor changes as Brise One Touch, and it took a 50% share of its local category.

Most of the fixes in that piece come down to process and credit. Procter & Gamble runs a version of an "adopt, adapt, create" sequence that requires a search for existing ideas before anyone builds an original one, and it hands out a "Proudly Found Elsewhere" award. For a software team, the equivalent can be one required section in any build proposal longer than a quarter, covering what already exists and what it would cost over three years. Someone outside the proposing team should read that section, since the people writing it are exactly the kind of settled group Katz and Allen studied.

Credit matters more than most engineering leaders admit. People rebuild partly because building is what gets noticed in performance reviews, so a team that praises a well-chosen integration as loudly as a clever internal tool changes what engineers volunteer to do. At SC Johnson, the chief executive officer [CEO] praised managers who used the company intranet to share outside ideas. In an engineering org, the same signal might be a slot at the quarterly demo for the integration that saved six weeks, presented by the engineer who found it.

- 1Not invented here
- 2PDF.js - Home
- 3In defense of not-invented-here syndrome

Summary

The tax is easy to miss because nobody sends an invoice for it. It piles up as maintenance hours and as senior attention pulled away from the parts of the product people actually pay for. Building still makes sense when the component is the reason customers buy from you, and Spolsky's line about core functions has held up for a quarter of a century. Everything else deserves a fair hearing against a version someone else already maintains. Today, pick the internal tool that ate the most engineering hours last quarter and write one sentence

naming the customer who'd choose your product because of it. If that sentence won't come, price the alternative before the next sprint planning.