



Rebuild or Refactor?

A 15-question scorecard for
product founders

Contents

How to use this scorecard.....	3
The 15 statements.....	4
Your score.....	6
Red flags that override the score.....	6
What each path involves.....	7
Before you commit: five questions.....	8
How SumatoSoft can help.....	9

How to use this scorecard

Your product works and customers pay for it, yet every release feels harder than the last. At some point the question becomes unavoidable: do you clean up what you have, or rebuild it?

This scorecard helps you answer that in about 10 minutes, without a consultant in the room. It looks at five areas where an outgrown codebase usually shows itself first: delivery speed, code health, architecture and scale, integrations and data, and cost and team.

1. Read each of the 15 statements and rate how often it is true for your product today: **0** = never, **1** = occasionally, **2** = often, **3** = constantly.
2. Answer for the product as it is now, not as it was a year ago or as you plan it to be.
3. If you can, fill it in together with your lead developer or CTO. Founders and engineers often see different symptoms.
4. Add up each section, then your total out of 45, and read your verdict.

The score is a starting point for a decision, not the decision itself. The "red flags" page lists situations that should move you toward a rebuild whatever your total is.

The 15 statements

Tick one box per statement. 0 = never, 1 = occasionally, 2 = often, 3 = constantly.

Delivery speed

Statement	0	1	2	3
1. Small features take weeks to ship, even when the change looks simple.				
2. Releases slip or get rolled back because something unrelated broke.				
3. We drop or reshape roadmap items because of what the code can handle.				

Delivery speed score: _____ / 9

Code health

Statement	0	1	2	3
4. Fixing one bug regularly creates another.				
5. Key parts of the system are understood by only one person, or by nobody still on the team.				
6. We have little automated test coverage, so every release needs manual checking.				

Code health score: _____ / 9

Architecture and scale

Statement	0	1	2	3
7. Performance drops noticeably as usage or data grows.				
8. Onboarding a new customer, tenant or location needs manual engineering work.				
9. Parts of our stack (framework, language version, libraries) are outdated or no longer supported.				

Architecture and scale score: _____ / 9

Integrations and data

Statement	0	1	2	3
10. Adding a new integration or API takes far longer than it should.				
11. Our data is scattered, inconsistent or hard to report on.				
12. The connected side of the product (apps, cloud sync, device data) is fragile or hard to change.				

Integrations and data score: _____ / 9

Cost and team

Statement	0	1	2	3
13. Hosting or maintenance costs grow faster than revenue.				
14. Hiring or onboarding engineers is hard because of our stack or codebase.				
15. The team spends more time maintaining the product than building new things.				

Cost and team score: _____ / 9

Your score

Section	Your score	Maximum
Delivery speed		9
Code health		9
Architecture and scale		9
Integrations and data		9
Cost and team		9
Total		45

Your verdict

Score	Verdict	What it means
0-15	Refactor	Your foundation is sound. The drag comes from specific problem areas that targeted clean-up can fix while the current architecture stays.
16-30	Partial rebuild	Some parts of the system are holding the rest back. Rebuild the modules causing the most drag and keep the core running.
31-45	Rebuild	The codebase now limits the business. Plan a staged rebuild, starting with a discovery phase rather than new code.

Look at your section scores too. A single section at 7-9 points to where the work should start, even if your total sits in the refactor band.

Red flags that override the score

Any one of these should push you toward at least a partial rebuild, whatever your total:

- A core framework, language version or third-party service you depend on has reached end of life, with no supported upgrade path.
- Known security vulnerabilities sit in components you cannot update without breaking the product.
- A new customer segment or market needs compliance or data-residency guarantees the current architecture cannot give.
- The only person who understands a critical part of the system is leaving, or has already left.
- Your product strategy is changing direction (new platform, new pricing model, multi-tenant) and the current design was never built for it.

What each path involves

Refactor (0-15)

Refactoring improves the code you have without changing what the product does. It is the lowest-risk path and can run alongside feature work.

- Add automated tests around the flows that make you money first: sign-up, billing, the core workflow.
- Find the few modules where most bugs come from and clean those up before anything else.
- Update outdated dependencies and document the parts only one person understands.
- Set a fixed share of each sprint for this work so it does not get crowded out.

Partial rebuild (16-30)

A partial rebuild replaces the worst parts of the system one at a time, behind stable interfaces, while the rest keeps running. Engineers often call this the strangler pattern.

- Map your modules by how much pain they cause and how much business value they carry.
- Pick one high-pain, high-value module and define clean interfaces around it.
- Build the new version next to the old one, move traffic over gradually, then retire the old code.
- Repeat module by module, so customers never face a big-bang switch.

Rebuild (31-45)

A rebuild is a new foundation for the product. Done well, it is staged and reversible; done badly, it freezes the roadmap for a year.

- Start with discovery: an architecture review, a clear list of what the new system must do, and what can be dropped.
- Plan data and customer migration before writing new code.
- Run old and new systems in parallel and move features or customer groups across in stages.
- Keep shipping small improvements to the old product during the rebuild, so customers see progress.

Before you commit: five questions

Whatever your verdict, answer these with your team before you start:

- ☐ Do we know which parts of the codebase cause most of the pain, or are we guessing?
- ☐ Is the current feature set the one customers actually pay for, or are we rebuilding things nobody uses?
- ☐ How much feature work can we slow down, and for how long?
- ☐ Who owns the knowledge of how the current system works, and how do we capture it?
- ☐ How will customer data and accounts move to the new or rebuilt parts without disruption?

How SumatoSoft can help

SumatoSoft builds and modernizes custom software for product companies: web and mobile apps, backends, cloud platforms and data pipelines, including the software side of connected products. We have helped founder-led teams decide between refactoring and rebuilding, and then deliver whichever path made sense.

Case study

[Insert the published ICP4 case study here: client and product, the problem with the original codebase, the path chosen (refactor, partial rebuild or rebuild), what was delivered, and the outcome. Use only figures stated in the published case study.]

Get a second opinion

Not sure your score tells the whole story? Book a free 30-minute codebase review. One of our senior engineers will go through your scorecard with you, ask about your stack and roadmap, and tell you honestly which path they would take.

Book your review: sumatosoft.com/contacts



Thank you