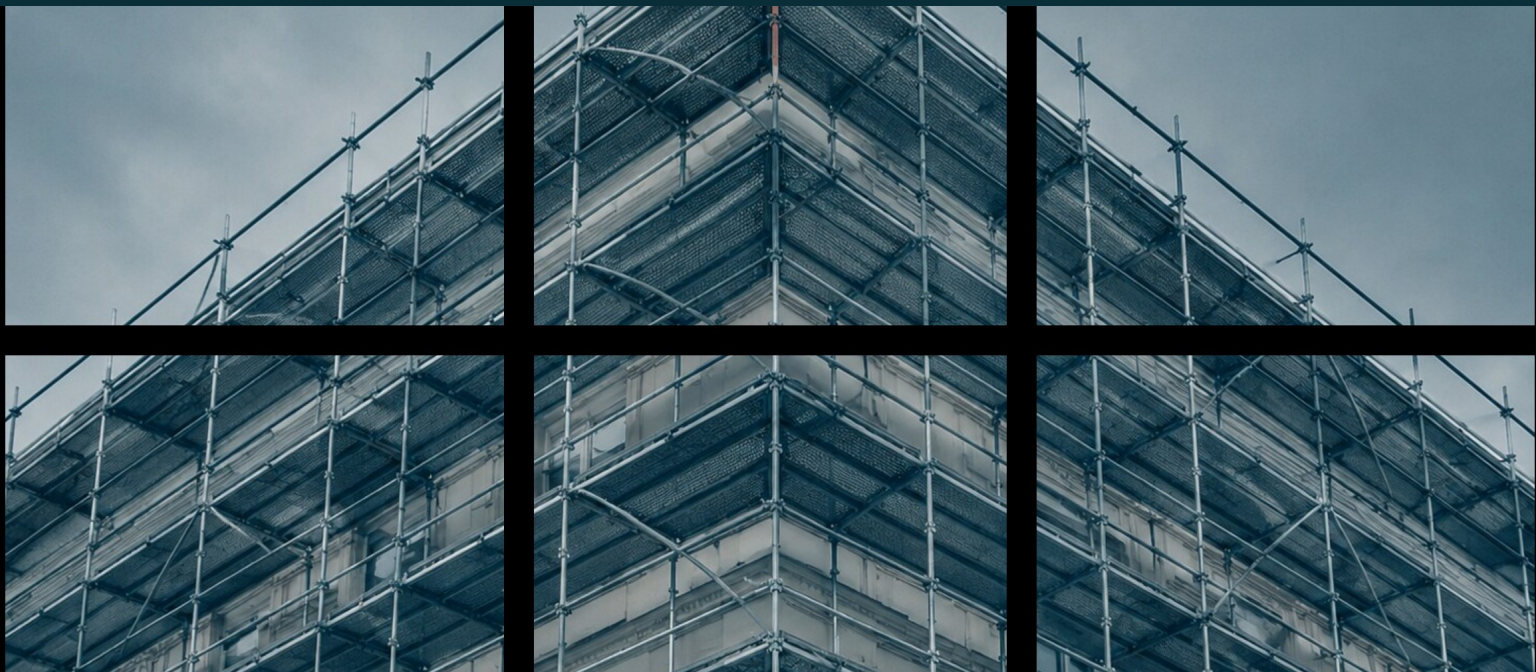


BEYOND THE PROJECT

The Software Was Never Finished: Product Thinking Before It Had a Name

A Practitioner's Perspective on Product Thinking in Software Delivery



Giovanni Leonardi

November 2011

The Software Was Never Finished: Product Thinking Before It Had a Name

Transformation · No. 209

Originally written November 2011 | Re-edited and re-rendered for the WhereBeyond Edition, 2025

“The requirement is a hypothesis, not a promise.”

The Word We Did Not Have Yet

We did not call it product thinking, because we did not yet have the word. In the software organisations of the last decade, the unit of work was the project: a thing with a start, a defined scope, a budget, and — crucially — an end. You initiated it, you delivered it, you closed it, and you moved the people on to the next one. The vocabulary we used every day — requirements, deliverables, sign-off, handover, go-live — all described a temporary endeavour that was meant to finish.

And yet, working inside that model, you would occasionally meet people who behaved as though something quite different were true. They did not treat the software as a deliverable to be completed and handed away. They treated it as a thing that was alive — that would keep changing, keep being used, keep mattering, long after the project that built it had been formally closed. They asked questions the project plan had no room for: not is it built to specification? but is it any good, and for whom? They were, in retrospect, doing product thinking. They just had no name for it, and neither did the rest of us.

Projects End; Products Do Not

The deepest assumption baked into the project model is that software is a thing you finish. This single assumption produces most of what goes wrong.

A project is optimised to reach its end. Every incentive points at closure: hit the date, land the scope, release the budget, disband the team, declare success. The moment the software goes live — which is to say, the moment it first begins to matter to the people using it — the team that understands it best is dispersed and the funding stops. We built the thing precisely up to the point where the interesting questions began, and then we walked away.

A project asks when is it done? A product asks is it working, for whom, and what should it become next? Almost everything follows from which of those two questions an organisation is really asking.

The people I am calling product thinkers had internalised the second question. They knew, without being able to articulate quite why, that the go-live was a beginning and not an end — that the real work of learning what the thing should be had barely started when the project declared victory.

The Tyranny of the Requirement

The other inheritance of the project era is the requirement. A requirement is a promise, captured early, when we knew the least, and then defended against change for the rest of the initiative. The discipline of the age was to manage scope — to freeze the requirements and treat any deviation as a failure of control.

Product thinking quietly inverts this. It treats the early specification not as a contract to be honoured but as the first and worst guess we will ever make about what is needed. The requirement is a hypothesis, not a promise. The job is not to deliver it faithfully; the job is to find out whether it was right, and to change it when it was not.

- The project person asks: have we built what was specified?
- The product person asks: have we built what was needed — and how would we even know?

These sound similar. They are not. The first can be answered by a document. The second can only be answered by contact with the people who use the thing — which is exactly the contact the project model is structured to avoid, because it happens after go-live, when everyone has gone home.

What the Product People Actually Did

If you watched them closely, the practitioners who had this instinct behaved in a recognisable handful of ways — none of which the methodology of the day rewarded.

1. They owned the outcome, not the output. They did not consider their job done when the feature shipped. They considered it done when the feature worked — when someone's actual problem was smaller than it had been. This made them hard to sign off, because sign-off is about output.
2. They said no. They understood that a product is defined as much by what it refuses to do as by what it does, and they fought to keep out of scope the things a project, hungry to please every stakeholder, wanted to let in.
3. They stayed close to the user. Not through a requirements document filtered up through three layers of business analysis, but directly — watching people struggle with the thing, which in that era you often had to do informally, almost apologetically, because it was nobody's defined role.
4. They thought in versions, not in releases. A release is a project milestone. A version is a point on a line that continues indefinitely. They planned as though there would always be a next version, because they assumed the product would outlive them — and it usually did.

The product thinkers were frequently regarded as difficult. They were difficult in the way anyone is difficult when they optimise for a goal the organisation claims to want but is not actually set up to measure.

Why the Organisation Resisted Them

It would be comforting to say these people were simply ahead of their time and eventually vindicated. Some were. But it is more honest to notice why the organisation resisted them, because the resistance was structural, not stupid.

Everything about how we funded, governed and staffed software work assumed the project frame. Budgets were annual and scope-based. Success was measured at closure. Careers advanced by delivering projects, not by tending products over years. A person who said this is never really finished was, from the organisation's point of view, refusing to let a budget be closed and a success be declared. They were not being visionary; they were being inconvenient. The system did not have a slot for what they were doing, so it treated them as poor project managers rather than as something the discipline did not yet have a name for.

Naming It

Lately the language is beginning to catch up. The idea of the persistent product — owned by a stable team, funded continuously, measured by the outcomes it produces rather than the scope it delivers — is starting to be spoken about openly, and it is starting to acquire a vocabulary. I welcome this, with one caution learned from watching other good ideas arrive: the danger is that we adopt the words and miss the substance. It would be entirely possible to rename every project manager a product owner, relabel every project a product, and change nothing at all about the annual budget, the scope freeze, or the incentive to finish and move on.

The substance was never the vocabulary. It was a way of seeing: that the software is not a deliverable but a living asset, that the specification is a guess and not a promise, that the work begins rather than ends at go-live. The people who saw that were doing product thinking long before we had the phrase. The task now is not to teach the phrase. It is to build organisations that let the people who already think this way stop being regarded as difficult — and start being regarded as right.