WHITEPAPER · AGENTIC AI FOR PRODUCT LEADERS

# The New PM Role

Three vectors, and only one of them is about tools


The conversation about AI and product management is stuck on the vector that matters least. Efficiency arrives for everyone on the same day. The two vectors that cannot be bought are the ones that decide whether you ship a faster version of today's product or a better one.

**Yoram Friedman, MD**
Physician and enterprise product leader · SAP, Walmart, Microsoft, regulated health technology
Drawn from the Agentic AI for Product Leaders series · agenticaiproductmanagement.com

THE ARGUMENT

# The job did not shrink. It moved, and most of the conversation is watching the wrong part.

Three things are changing for product managers at once, and they are usually discussed as one. Separating them is most of the work, because they have completely different half-lives.

**Efficiency.** AI compresses the day-to-day: translation between audiences, knowledge synthesis, drafting, restating, prototyping. This is where almost all of the attention is.

**Judgment.** A product mindset is what turns an agent from a demo into a product that is safe to run. This is where almost none of the attention is, and it is the bulk of the actual work.

**Collaboration.** The artifacts that pass between product, engineering and design have changed shape, which means the way the three of them work together has changed with it. This is the least written about of the three and the most immediately useful.

> There is a fast test for which vector you are thinking about. If the comparison to Excel feels right, you are on vector one. Excel absorbed an enormous amount of analyst busywork and it never decided anything. The vectors where that comparison breaks are the ones that are actually your job now.


Figure 1. What moved. The layer that filled the calendar is the layer the tools absorb. The layer underneath it, which most product managers were exercising in the margins, becomes the job. The total does not shrink. Schematic, not measured.

BEFORE THE VECTORS

# Four waves changed the vehicle. This one changed the constant.

For twenty years every generation of enterprise software felt like a disruption, and every one of them left the same thing untouched.

Client and server moved the screen off the mainframe, and a trained user still read it and decided. Service-oriented architecture had systems talking to each other, and a trained user still approved what they did. Cloud and mobile changed where and when the work happened, and the user was still there, reading and deciding.

Every one of those was an efficiency wave. They changed the vehicle. The person at the end of the path never changed, which is exactly why the tooling frame kept working for four decades, and why the comparison to spreadsheets holds so comfortably for all of them.

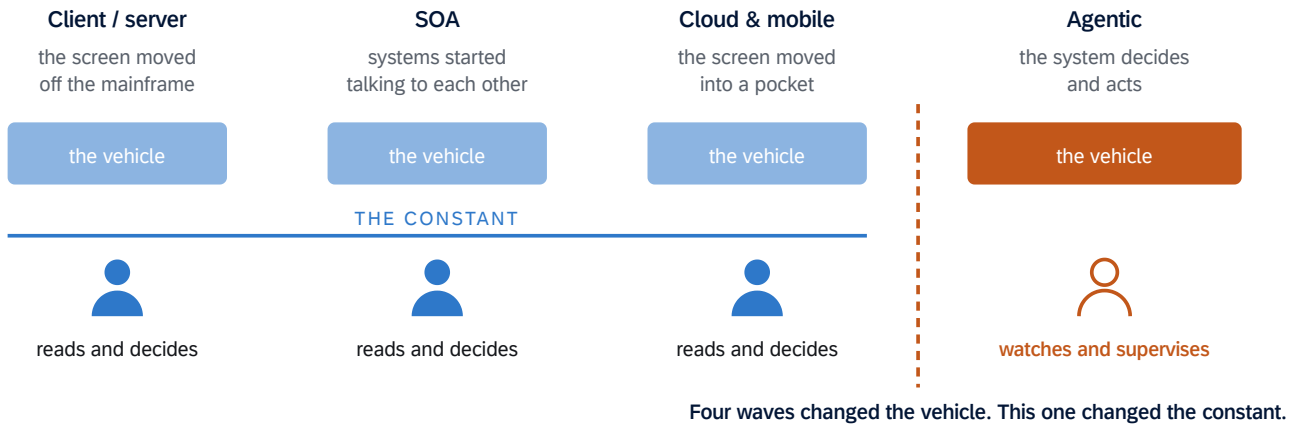


Figure 2. Every wave moved the vehicle and left the person at the end of the path in place, reading and deciding. This is the first one where that person changes role. Schematic, not measured.

This wave removes the person from the path. That is not an increment on the previous four. It is the first time the constant moved, and it is why the artifacts we trusted all break on the same question.

The user story puts a person at the center: as a user, I want, so that. The journey map draws the path a human walks, step by step. Both of them assume somebody is walking it. Ask whether you can replace the persona with a model and the whole apparatus wobbles, not because the apparatus was badly built, but because it was built around a constant that has stopped being constant.

Hold that alongside the three vectors, because it explains the asymmetry between them. The efficiency vector is this wave behaving like the previous four. The other two are this wave doing the thing the previous four never did.

VECTOR ONE

## Efficiency, and why it will not move the needle

This is the vector everyone is talking about: tools, prototyping, faster synthesis, new skills. It is real. It is also the one that changes the least about what you ship.

Start with what it actually absorbs. A large language model is, at its core, a translation engine: it takes intent in one form and renders it in another. That landed directly on the part of the product manager's job that filled the most calendar hours. Writing the requirement. Drafting the story. Restating the same need in the language of engineering, then again in the language of the executive, then again, more slowly, for the person who missed the first three times.

That was the how of the job. It is the single thing a language model does most naturally, and it is being absorbed faster than any other part.

### The Excel comparison, taken seriously

Fifteen years ago a comparable thing happened to analysts, and the comparison is worth making properly rather than as reassurance.

Spreadsheets absorbed an enormous amount of analytical busywork. They did not eliminate analysts. What they did was end the career of the analyst whose value was the mechanics, and promote the one whose value was knowing which question to ask. And here is the part that matters for the current moment: spreadsheets made every company's analysts faster and made no company better at analysis than its competitors. The gain was universal, so it was not an advantage. It was the new floor.

> Efficiency that arrives for everyone on the same day is not an edge. It is a baseline, and the only question is how quickly you stop congratulating yourself for reaching it.

### The part that is not neutral

If vector one merely failed to differentiate, it would be harmless. It does something worse than that. It lowers the cost of building the wrong thing.

A coding agent will produce a working prototype of almost anything you can describe in an afternoon. That is useful when the prototype exists to settle a question. It is dangerous when it exists to be shown, because the moment a prototype demos well, the go or no-go decision leaves the room and does not come back. Nobody kills a thing that works on a screen.

So the efficiency vector, on its own, produces more candidate products reaching a demo, each one arriving with its decision already made. Faster is not better here. Faster is the same product sooner, and occasionally the wrong product with a stronger alibi.

### What the market is already doing about it

The commoditization is visible in hiring rather than only in tooling. Tom Verrilli, the chief product officer at Whatnot, has described building a product organization on the premise that they regret product management exists: not that PMs are useless, but that the industry started hiring them by ratio rather than by need, and that a PM added to every pod removes a rep from the engineers and designers who could have made the decision themselves. His team has taken thirty-one thousand eight hundred and thirty-two applications for product roles over two years and hired one.

He is also specific about what is trending down in interviews: candidates whose specialty was driving alignment and stakeholder management. The skill that got commoditized is fast, accurate translation. The skill that did not is the one underneath it.

Which is where the Excel comparison stops working, and the comparison's failure is the most useful thing about it. A spreadsheet computed exactly what you asked, the same way every time, and never made a judgment call. An agent decides. That single difference is the seam between vector one and everything that follows.

VECTOR TWO

## Judgment, which is the bulk of it

Translation was never the job. It was the visible part of the job, the part that filled the day and made it look like the product manager was a relay between functions. Underneath it was something the translation was carrying, and that thing is not only surviving the automation. It is becoming the most consequential work on the team.

Four questions make up that work, and none of them has a tool. Each has an owner.

ONE

**Does this problem deserve an agent at all?** The comparison is never against doing nothing. It is against the best available alternative, which is often deterministic automation, a constrained non-agentic feature, or a human with better tooling. For routine routing, an agent runs a hundred to two hundred times the cost per task of ordinary automation, and the differential is structural rather than marginal. The answer "we built an agent because we could build an agent" is now the most expensive sentence in enterprise software.

TWO

**Where does its authority stop?** Autonomy is a ladder, not a switch, and every product sits on a rung. Placing it accurately, and naming the single demonstrated competence that would justify moving it up one, is a product decision that engineering cannot make for you and a platform will not make for you.

THREE

**What does good mean when the output is probabilistic?** Run the same case twice and the agent, drawing on different retrieval and different context, may answer differently. So correctness is a distribution rather than a check, and defining it is authorship, not testing.

FOUR

**Who answers for the people who never appear in your analytics?** The people inside an agent's error rate are, almost by definition, the ones least able to make you notice. They are frequently not your users. Nobody escalates on their behalf because nobody is assigned to.

### Try this before your next review

Take one prompt and run it ten times in the same session, unedited. Something trivial works best: *in twenty words or less, describe the difference between apples and oranges*.

Do not cherry-pick. Ten runs, ten answers. If a trivial prompt will not repeat itself, then the pass-or-fail acceptance criterion you wrote for your agent's judgment is measuring exactly one sample of a distribution you have never looked at. The exercise takes four minutes and it changes how a room argues about testing.

Notice what these have in common. Every one of them is a decision that has to be made before the tools are useful, and every one of them is invisible in a demo.

## Agent is a noun. Agentic is a dial.

One of those is what you buy. The other is what your system does, and only you can set it. Every vendor sells you the noun: a model, a set of tools, a loop, a component with a name on an invoice. Nobody sells you the dial, because the dial is the answer to how much of the loop your system closes without a person in it, and that is a judgment about your domain, your risk and your users.

Teams that treat the noun as the decision end up discovering their position on the dial after launch, by observing it. Teams that treat the dial as the decision place the product on a rung deliberately and name the one demonstrated competence that would justify moving it up. Same procurement, entirely different product.

## The dial has rungs, and the judgment scales with them

Autonomy is not one setting, it is a ladder, and the human role changes at every rung. At rung one the system suggests and the person is a chooser. At rung two it drafts and they are an editor. At rung three it acts with approval and they are an approver. At rung four it acts with oversight and they are a reviewer. At rung five it acts autonomously and they are a supervisor.

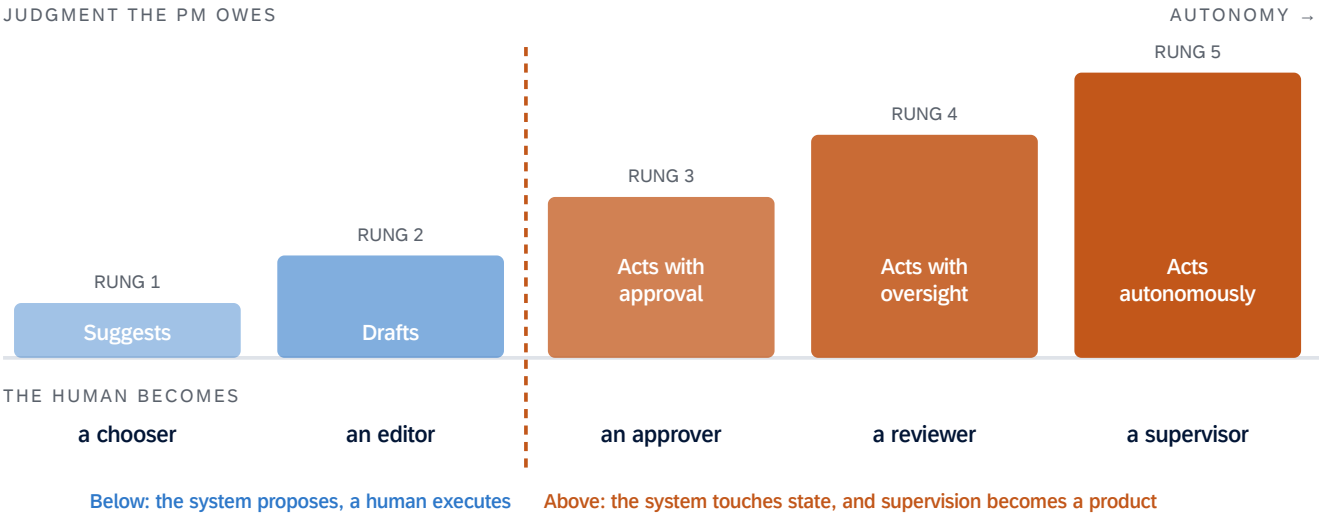


Figure 3. The autonomy ladder, and the reason the second vector is not a fixed quantity. A rung is earned by evidence at the rung below, never scheduled. A product is not on one rung; its tasks are. Schematic, not measured.

The reason to put this in a paper about the PM role rather than only in a paper about supervision is that it explains why the second vector is not a fixed quantity. The judgment you owe is a function of the rung. At rung one a bad output is ignored by a person who was

always going to decide anyway. At rung five a bad output is an action that already happened. Same model, same prompt, same team, and a completely different amount of product work owed.

Which is where most of the damage occurs. Teams climb the ladder incrementally, feature by feature, because each individual step looks small, and they climb without re-paying the judgment the new rung requires. Nobody decides to move from approver to reviewer. It happens because approvals were slow.

There is a line on that ladder worth marking explicitly. Below it, the system proposes and a human executes; mistakes are recoverable by construction, and this is the governance posture enterprises have run for two decades on workflow and process automation. Above it, the system touches state, and supervision stops being a review step and becomes a product with its own requirements.

One nuance keeps teams honest here. Low rungs are automation-like in governance posture, not in nature. A suggests-only model is still probabilistic and the suggestion can be confidently wrong. What makes the low rungs safer is the human hand between the output and the world, not determinism. Over-governing tends to stop below the line. Under-governing gets named above it.

### Why a tool would not have caught it

When researchers tested a general AI assistant on real emergency-department triage, it under-triaged about fifty-two percent of genuine emergencies, routing people who needed an emergency department toward wait-and-see care.

The system was not broken in any way an evaluation suite would flag. It produced fluent, well-formatted, confident dispositions. It had been built against the intake form, the structured fields, rather than against what an experienced triage nurse actually listens for: the cadence, the thing the caller mentions as an aside, the wrongness underneath the words.

No amount of vector one would have found that. A faster prototype would have reached the demo sooner. Only someone with domain judgment sees that a plausible answer is the wrong answer, and only a product decision puts that person in the loop before launch rather than after the incident.

This is the whole argument for why efficiency alone does not move the needle. The tools make it cheaper to build what you already know how to describe. They do not tell you whether it should exist, where it must stop, what good means, or who carries the consequence. An organization that invests only in vector one gets to the same product faster. An organization that invests in vector two gets to a different product.

VECTOR THREE

## Collaboration, because the artifacts changed for everyone

The most concrete of the three, and the least discussed. The unit of work that passes between product, engineering and design is no longer the same object, which means the working relationship built around that object has to change too.

### What engineering is now asking for

The clearest way to hear the shift is to listen to what engineering asks you for. The request used to be a screen and the behavior behind it. The request now is the boundary: intent, policy, risk tolerance, escalation rules, and an acceptance criterion a test harness can actually check. Give me the boundary, not the screen.

Most product organizations have not changed what they hand over. They are still shipping personas, flows and stories into a build process that needs authority, limits and graded cases. That gap is not a communication problem and it will not be closed by a better template. The unit of work itself changed, from what the user does to what the agent may do.

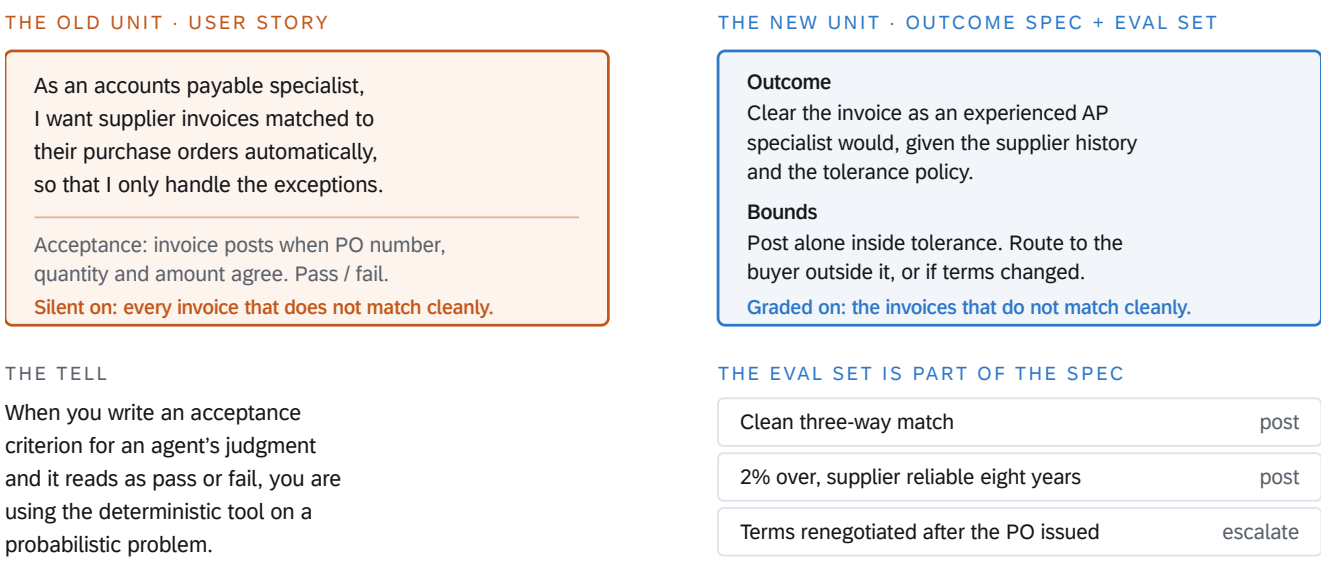


Figure 4. The work-unit shift. The story describes the path that needed no human. The outcome spec states the intent and the bounds, and the eval set carries the cases that did, each paired with the call an experienced person actually endorsed. It is not a better-written story. It is a different primitive.

The tell is easy to spot in your own documents. When you find yourself writing an acceptance criterion for an agent's judgment and it reads as pass or fail, you are using the deterministic tool on a probabilistic problem. The story format was built for software that does what you specified. It has no field for the invoice that is two percent over from a supplier who has been reliable for eight years, and that case is the only reason a human was in the loop at all.

## The new engineering conversation

Two things moved on the engineering side and both change what you owe each other.

Generation got cheap and verification did not. A review of four hundred and seventy real pull requests found AI-generated changes carried roughly 1.7 times the issues of human changes, with logic and correctness errors up about seventy-five percent and error-handling gaps roughly doubled. The pattern has a name, the eighty percent problem: the agent nails the well-represented part and degrades across the remainder, which is exactly where the edge cases and the security assumptions live. The cost did not disappear. It moved from writing to reviewing, and it moved onto a team that did not sign the business case.

And enforcement is not yours. You decide where the boundary sits; the architect decides whether a wall stands there. A boundary written into a brief as prose the agent will read is a boundary the agent can reason its way past. A boundary written as a requirement with a number on it is one someone has to close. The failure lives in the hand-off, not in either person's work, which is why that hand-off now needs a check in the middle rather than a hope.

### The new design conversation

Design gained a second user and it is often the same person. As a product climbs the autonomy ladder, the person who used to do the work becomes the person who watches the work being done. That person needs a product: the queue, the approval moment, the intervention, the record. A confirmation box asking "proceed?" transfers liability without transferring understanding.

Which makes the approval moment a design artifact rather than a dialog, and makes supervision a surface with requirements rather than something operations absorbs after launch.

PUTTING THE THREE TOGETHER

## Which vector are you being paid for?

The three vectors are not equal, and treating them as three equal thirds is how a real shift gets flattened into a list.

One of them is a tooling story. Two of them are a job-redefinition story. Vector one has the shortest half-life of anything in your current role, because it is universally available, improving monthly, and yours only until the next release reaches everyone else. Vectors two and three cannot be bought, cannot be installed, and do not arrive on a release schedule.

That suggests a blunt question worth asking about your own week. If a large part of it is spent on translation, restatement, synthesis and status, you are being paid for the vector with the shortest half-life. That is not a reason for alarm and it is a reason to move deliberately, because the work does not migrate on its own and nobody will hand it to you.

The honest answer, though, is that there is no single right mix, because the mix is a function of the rung. On a rung-one product the work really is mostly efficiency, and that is not a failure of ambition; a suggestion a person was always going to weigh needs speed more than it needs governance. Climb to rung three and judgment starts to dominate, because the system now touches state. Climb to rung five and collaboration dominates too, because the artifacts, the enforcement and the supervisory surface all have to be built by people who are not you.

> The vector you should be paid for is set by the rung your product sits on, not by your title. The expensive failure is a team operating at rung four while still being measured, staffed and rewarded for rung one.

That failure is common and it is quiet, because nothing announces the climb. Approvals get slow, so a gate is relaxed. A workflow gets automated end to end because the pieces already existed. The product moves up a rung and the operating model does not follow, and the gap between them is where the incident eventually happens.

### Three questions, one per vector

**Efficiency.** What did I do this week that a competent agent could have done, and what did I do with the time I got back? If the honest answer to the second half is "more of the first half," the gain has been absorbed rather than reinvested.

**Judgment.** On the agentic product closest to shipping, can I state its rung on the autonomy ladder, the one competence that would justify moving it up, and the name of the person who answers for someone the agent harms who is not our user?

**Collaboration.** Is my next specification an outcome with bounds and a graded eval set, or a story with pass-fail acceptance criteria? And has an architect confirmed that a wall stands where I drew the line?

None of those three questions has a tool that answers it, which is the point. They are the part of the job that survived, and they were always the part that mattered.

WHAT THIS IS FOR

## An agentic enterprise is not built by faster product managers.

It is built by product managers who can decide what should be autonomous at all, where its authority stops, what good means when the output is probabilistic, and who answers when it is wrong. Tools make the building cheap. Judgment is what makes the result better than what we already have, and no release note is going to deliver it.

---

**Sources and status.** Drawn from the *Agentic AI for Product Leaders* series by Yoram Friedman: *Agentic AI for Busy Product Managers* (second edition) on the autonomy ladder, suitability and the cost comparison against traditional automation; *Why Agentic AI Products Fail* on the bridge-operator framing, the work-unit shift, the two briefs and the emergency-department triage case; *The Agentic AI Team* on what the translation layer was carrying, on AI-generated code quality, and on the seam between the

product manager who decides a boundary and the architect who enforces it; and *The Agentic AI Practitioner* (forthcoming, September 2026) on the prototype that removes the go or no-go decision from the room. The pull-request finding is a review of four hundred and seventy real pull requests, reported with the Georgetown code-security review alongside it. The emergency-department triage figure is from published research on a general assistant tested against real triage cases. Tom Verrilli's remarks on product-management hiring are from his appearance on Lenny Rachitsky's podcast and are quoted as his position, not endorsed in full. Figures 1, 2 and 3 are schematic illustrations of the argument, not plots of measured data. The invoice-matching example in Figure 4 is an illustrative composite.