
Module 1: Fundamentals of Product Management

What a product is, what a PM actually does, how a PM thinks and decides, and how product work is organised.



AUTHOR

Varun Kumar

Product Management Master Handbook

v0.1 · 23 October 2026

Built from the 55-lecture course study guide and extended to the 12-module Product Core curriculum. Hypothetical examples are labelled; no real company results are invented.

Contents

1.1 How to read this module	3
1.2 What is a product?	4
1.3 What is Product Management?	5
1.4 How a PM thinks: trade-offs, evidence and outcomes	7
1.5 PM versus adjacent roles	8
1.6 PM specialisations and career paths	10
1.7 Product lifecycle versus product development lifecycle	11
1.8 Product operating models	13
1.9 Vision, mission and strategy (overview)	14
1.10 Product-market fit	15
1.11 Customer value versus business value	16
1.12 Output versus outcome	17
1.13 Product discovery versus product delivery	19
1.14 Stakeholder management (introduction)	19
1.15 Product sense and product thinking	20
1.16 Growth model and customer relationship: PLG versus SLG	21
1.17 B2B versus B2C product management	22
1.18 Platform products versus feature products	22
1.19 Common Product Management mistakes	23
1.20 Source-flagged misconceptions (corrected)	23
1.21 Practice and model answers	24
1.22 Interview perspective	25
1.23 Key takeaways	25
1.24 Check your understanding	27
1.25 Recommended resources	28

MODULE 1

Fundamentals of Product Management

What a product is, what a PM actually does, how a PM thinks and decides, and how product work is organised.

≈ 6 h study

1.1 How to read this module

Each topic is taught at three depths. Look for the coloured boxes:

Level 1 · Beginner

The idea in plain English, no jargon.

Level 2 · Working PM

How you use it in a sprint, a roadmap review or a stakeholder meeting.

Level 3 · Senior PM

Strategy, organisational constraints, money and risk.

From your course notes

Content preserved from the 55-lecture study guide (page references in the module header). Where the guide itself flagged a lecture statement as inaccurate, the corrected version is taught here.

Hypothetical example

Invented numbers for teaching. They are **not** real company data.

1.2 What is a product?

A **product** is anything a person or organisation uses to achieve a goal, delivered in a repeatable way. For a PM the useful definition is:

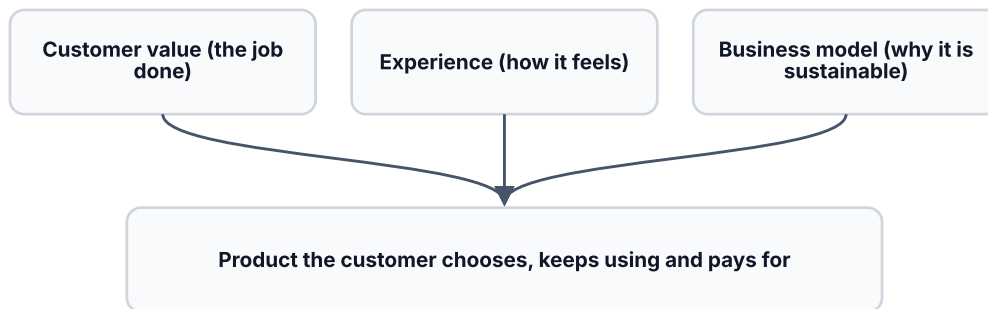
A product is a bundle of **value** (the job it does for someone) and **experience** (how it feels to get that job done), sustained by a **business model** (why the organisation can afford to keep providing it).

Level 1 • Beginner

A food-delivery app is a product. Its value is "I get lunch without cooking or leaving my desk." The experience is browsing, ordering, paying, tracking and receiving. The business model is the commission and fees the company earns. If any of the three is broken, the product fails: great value with a terrible experience loses customers; a lovely experience with no business model runs out of money.

Kinds of product you will meet

Type	Example (illustrative)	What the PM mainly optimises
Consumer app	A food-ordering or ride-hailing app	Activation, repeat usage, trust
B2B SaaS	A team-collaboration or CRM tool	Adoption inside accounts, renewals, expansion
Marketplace	Buyers and sellers, riders and drivers	Liquidity: balanced supply and demand
Platform / API	A payments API used by other developers	Developer experience, reliability, ecosystem
Internal tool	An operations dashboard	Employee productivity, error reduction
AI product	A support assistant	Answer quality, safety, cost per resolution



A product is value + experience + business model. Removing any one input produces a recognisable failure mode — useless but pretty, useful but painful, or loved but unprofitable.

1.3 What is Product Management?

Product management is the discipline of deciding **which problems are worth solving, for whom, and how the team will know the solution worked**, then guiding the team to deliver it. The PM does not “own” the product alone; they own the *clarity of the problem and the quality of the decisions*.

From your course notes (Lecture 2)

- Products deliver value by solving user needs; a feature should trace back to a real need.
- PMs balance **user needs, technical feasibility and business goals**, and collaborate with designers, engineers and stakeholders. *This does not automatically give a PM formal authority over everyone.*
- Specialisations differ by company: **Technical PMs** (platforms, infrastructure), **Data PMs**, **Growth PMs** (acquisition → revenue), **Business PMs**. An **AI-enabled PM** uses AI tools in their work; an **AI PM** manages a product whose functionality depends on AI. One person may do both.
- Product-market fit is not proven simply because a market exists or some demand exists.

1.3.1 The three circles — and the fourth risk

A good product sits where three things overlap: **desirable** (people want it), **feasible** (the team can build and run it) and **viable** (it works for the business). Marty Cagan frames the same idea as four risks a PM must retire before and during building: **value** risk, **usability** risk, **feasibility** risk and **business-viability** risk.

Risk	Question it asks	Typical evidence	Main partner
Value	Will anyone choose and use this?	Interviews, prototype tests, fake-door tests, retention	Users, design, data
Usability	Can they figure out how to use it?	Usability tests, task success rate, time-on-task	Design, research
Feasibility	Can we build and run it with our time, tech and skills?	Engineering spikes, architecture review, cost estimates	Engineering
Viability	Does it work for the business — legal, financial, brand, ops?	Unit economics, legal and compliance review, GTM readiness	Finance, legal, sales, ops

The four product risks and who helps you retire them. Discovery work exists to retire these risks cheaply, before the expensive part (building) begins.

1.3.2 What a PM is responsible for

1. **Problem definition** — who has the problem, how often, how painful, what they do today.
2. **Strategy and focus** — saying “not now” so the team works on the highest-value problem.
3. **Discovery** — talking to users, reading data, running cheap experiments.
4. **Requirements and alignment** — turning understanding into artefacts the team can build from (PRDs, user stories, acceptance criteria).
5. **Delivery partnership** — removing blockers, making scope trade-offs, protecting quality.
6. **Launch and learning** — go-to-market coordination, metrics, post-launch review.
7. **Stakeholder communication** — keeping leadership, sales, support, legal and operations informed.

Level 2 · Working PM

On a typical week you might: review last week's funnel, run two customer calls, refine the next sprint's stories with an engineer, answer a sales request with data about why you won't build it this quarter, and update a one-page decision note. The through-line is *decisions made with evidence*, written down so others can challenge them.

Level 3 · Senior PM

Seniority is less about more features owned and more about **bigger ambiguity and cost of being wrong**: choosing which market to enter, how to price, whether to build vs buy vs partner, how to sequence a multi-team programme, and how to say no to a powerful stakeholder without losing the relationship.

1.4 How a PM thinks: trade-offs, evidence and outcomes

PMs make decisions with incomplete information. Good ones follow a loop, not a gut feel:



The PM decision loop. The loop repeats at every scale — a button label, a sprint, a quarterly bet. The step people skip most often is "Learn".

Trade-offs you will make daily

Tension	Example	Question to ask
Speed vs quality	Ship a payment retry now or harden the flow first?	What is the cost of a failure here — money, trust, regulation?
Short-term vs long-term	Quick revenue feature vs platform investment	What does this make cheaper or harder next quarter?
New users vs existing users	Onboarding polish vs power-user requests	Which segment drives the metric we are accountable for?
Breadth vs depth	Ten shallow integrations vs one excellent one	Where do customers actually get stuck today?
Customer ask vs customer need	"Add an export button" vs "I need to share results with finance"	What job is behind the request?

Worked example · reading a request as a problem

A sales lead asks for “a PDF export on the dashboard.” The PM asks *what happens after the PDF is created?* and learns the customer pastes the numbers into a monthly slide for their boss. Options now include a scheduled email summary, a shareable link, or a PowerPoint export — each has a different cost and risk. The PM picked the problem (“managers need to share monthly numbers”), not the solution.

Interview perspective

When asked “how do you make product decisions?”, describe the loop above, give one real example with a trade-off you explicitly named, and say what you would measure to know you were right. Interviewers listen for *evidence, trade-offs and learning*, not frameworks by name.

1.5 PM versus adjacent roles

Role boundaries differ by company. These are **typical** emphases, not rules; always ask “who makes the final call on X?” in a new team.

Role	Core question	Typical outputs	Where it overlaps with PM	How they differ
Product Manager	Which problem should we solve, for whom, and how will we know it worked?	Strategy, roadmap, PRD, success metrics	–	Owens the why and the outcome
Project Manager	How do we deliver this scope on time and within budget?	Plan, schedule, RAID log, status reports	Delivery coordination	Project is temporary with a fixed end; product is ongoing and judged on outcomes
Product Owner (Scrum)	What is at the top of the backlog and is it clear enough to build?	Ordered Product Backlog, acceptance criteria	Backlog and prioritisation	A Scrum accountability; in some firms the PM does it, in others a separate person with a narrower, delivery-facing scope
Business Analyst	What exactly does the business process need?	Process maps, requirements, UAT cases	Requirements detail	Deep on current-state process and rules; less on market and strategy
Product Designer	How should it work and feel for the user?	Flows, wireframes, prototypes, design system	Discovery and solution shaping	Owens the experience craft; PM owns the business and prioritisation context
Product Marketing Manager	How do we position, message and launch it?	Positioning, messaging, launch plan, sales enablement	Go-to-market	Owens market narrative and launch; PM owns the product

Role	Core question	Typical outputs	Where it overlaps with PM	How they differ
Engineering Manager	How do we build it sustainably with a healthy team?	Technical plan, staffing, quality, team health	Delivery and scope trade-offs	Owens people and technical execution; PM owns problem and priority

PM compared with neighbouring roles (typical, not universal). Most real friction comes from unclear decision rights, not from the job titles. Agree who is Responsible, Accountable, Consulted and Informed for each type of decision.

From your course notes (Lecture 12)

Product Managers, Product Owners and Business Analysts may divide discovery, requirements and prioritisation differently across companies. Product management focuses on **value and outcomes**; project and program management emphasise **delivery coordination** at different scopes. Exact responsibilities vary.

Common confusion

“PM = Product Owner” is true in some teams and false in others. In Scrum, the Product Owner is **one person** accountable for maximising the value of the product and managing the Product Backlog. A PM may *be* that person, or may work above the team setting strategy while a PO handles day-to-day backlog detail.

1.6 PM specialisations and career paths

From your course notes (Lecture 2)

A possible progression is **Associate PM → PM → Senior PM → Group PM / Director → VP Product → Chief Product Officer**. Domain expertise, communication, analytical thinking and case-study practice can support a transition into PM. Career paths vary.

Specialisation	Focus	Typical metrics
Growth PM	Acquisition, activation, retention, monetisation experiments	Conversion, retention, CAC, LTV
Technical / Platform PM	APIs, infrastructure, developer tools	Reliability, latency, API adoption, developer satisfaction
Data PM	Data products, pipelines, analytics tooling	Data quality, adoption, time-to-insight
Business / Strategy PM	P&L ownership, partnerships, pricing	Revenue, margin, contribution
AI PM	Model-powered features, evaluation, safety	Answer accuracy, cost per task, escalation rate

1.7 Product lifecycle versus product development lifecycle

Two different lifecycles are easy to mix up.

- **Product Life Cycle (PLC)** — how a product's *market life* evolves: **introduction** → **growth** → **maturity** → **decline**. It guides strategy: invest to find fit and acquire in introduction; scale and defend in growth; optimise margin and extend in maturity; decide to harvest, reinvent or retire in decline.
- **Product Development Lifecycle** (often the **SDLC** for software) — how a *piece of work* moves from idea to release: **discover** → **plan** → **design** → **build** → **test (QA/UAT)** → **release** → **learn**.



Product Life Cycle (market view). Classic PLC is a model, not a law — good product management can renew a mature product (new segments, new jobs) before decline sets in.



Product development lifecycle (work view). In practice these overlap and repeat; the arrows show emphasis, not a one-way waterfall.

1.7.1 Waterfall, Agile, Scrum, Kanban, SAFe — what to know as a PM

From your course notes (Lecture 12)

- **Waterfall:** sequential phases; late changes are costly. **Agile:** iterative delivery and learning from feedback; the best choice depends on context.
- **Scrum** organises work into **sprints**. *A sprint is at most one month; it does not have to be two weeks.* Scrum defines accountabilities, events and artefacts — read the official Scrum Guide.
- **Kanban** manages and improves *flow*, including **limiting work in progress (WIP)** — it is not simply a mix of Scrum and XP.
- **SAFe** coordinates work across many teams; **XP** emphasises engineering practices. Scrum and SAFe are not the only Agile approaches, and Agile existed well before the mid-2010s.
- **UAT** checks whether the solution meets acceptance needs; it cannot guarantee zero production issues.
- **Definition of Ready** (criteria a team may use before starting work) and **Definition of Done** (shared quality criteria for completed work) are different things.

Approach	Cadence	Best when	Watch out for
Waterfall	Phases in sequence, one big release	Requirements are stable and well known; heavy compliance sign-offs	Late discovery of wrong assumptions
Scrum	Fixed-length sprints (≤ 1 month) with planning, review, retrospective	Product learning and frequent, inspectable increments	Ceremony without outcome focus; velocity treated as productivity
Kanban	Continuous flow with WIP limits	Support, maintenance, unpredictable arrivals	Never-ending board with no goals
SAFe / scaled	Planning increments across many teams	Large programmes needing coordinated dependencies	Process overhead; teams lose autonomy
Dual-track Agile	Discovery and delivery run in parallel	Product bets with high uncertainty	Discovery becoming a separate silo from delivery

Delivery approaches at a glance. Pick the lightest process that gives you fast feedback and manageable risk. There is no universally best choice.

Level 2 · What you do in Scrum as a PM or PO

You keep an **ordered Product Backlog**, write items so the team can ask good questions, attend **Sprint Planning** to propose a Sprint Goal, answer questions during the sprint, and use the **Sprint Review** to gather stakeholder feedback. You do not tell developers *how* to build it, and you do not treat story points as individual productivity.

1.8 Product operating models

An **operating model** is how the organisation turns ideas into shipped product: who decides, how teams are formed, how funding and priorities flow.

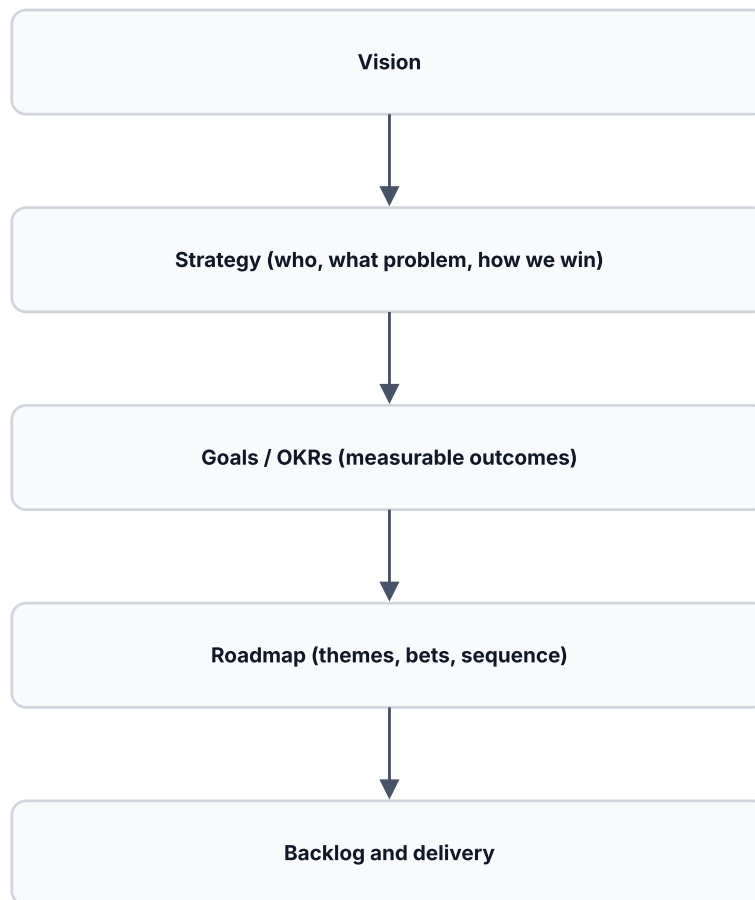
Model	How it works	Strengths	Failure mode
Project-based / feature factory	Business hands requirements to delivery; success = “shipped on time”	Predictable scope and cost	Output without outcome; teams don't own results
Product-based (project-to-product)	Long-lived teams own a product area and its metrics	Continuity, learning, accountability	Needs strong product leadership and trust
Empowered product teams	Teams get problems to solve with measurable outcomes , not solutions	Faster learning, higher morale, better ideas	Requires capable PM/design/engineering trio and clear strategy
Platform + feature teams	Platform team builds reusable capabilities; feature teams assemble them	Reuse, consistency	Platform becomes a bottleneck or builds what nobody uses
Matrix / functional	Reporting by function; work organised by project	Deep craft	Hand-offs, diluted ownership

Level 3 · Senior PM

Operating-model problems often masquerade as product problems. If every initiative arrives as a solution from the top and teams are measured on delivery, no amount of discovery skill will help. Senior PMs diagnose the model, then change *incentives and decision rights* — e.g., replacing a feature roadmap with outcome-based goals and quarterly reviews of learning.

1.9 Vision, mission and strategy (overview)

- **Mission** — why the organisation exists and what it does.
- **Vision** — the future you want to create for users.
- **Strategy** — the *choices* about whom to serve, which problem to solve and how to win sustainably (including what *not* to do).
- **Roadmap** — a communicated plan of outcomes and likely work over time; not a promise of dates.



From vision to outcome. Each layer constrains the one below. If a backlog item cannot be traced upward to a goal, question it. Module 2 develops strategy in depth.

From your course notes (Lectures 10, 13; Extra topic 1)

A vision need not serve every person on earth, and it can evolve. A roadmap can be **feature-based** or **goal-based**; build it with engineering, design and stakeholders; treat it as an evolving plan, not a guarantee.

1.10 Product–market fit

Product–market fit (PMF) means a product satisfies a real market need strongly enough that customers keep using, paying for and recommending it. Marc Andreessen's often-quoted framing is being "in a good market with a product that can satisfy that market."

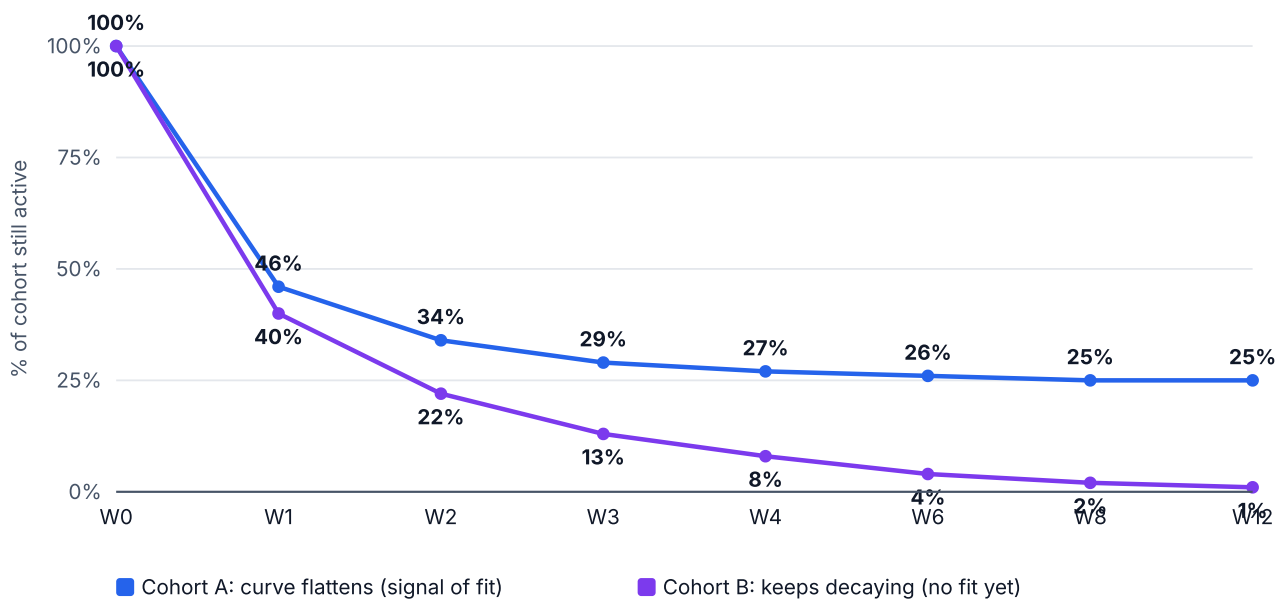
Signals of fit (use several, never one):

- Retention curves that **flatten** rather than decay to zero (see Module 7).
- Growing share of new users arriving by word-of-mouth or referral.

- The **“very disappointed” survey test** (popularised by Sean Ellis): if roughly **40%+** of active users say they would be *very disappointed* if the product disappeared, that is a commonly used signal of fit. It is a heuristic, not a law.
- Customers pulling the product faster than you can serve it; sales cycles shortening.
- Unit economics that work at the margin (Module 7).

Not evidence of fit

A large TAM, a polished prototype, a high survey intent score, or “people said they'd use it.” The course notes make the same point: a survey preference is not proof of purchase, and picking the largest segment alone does not prove fit.



Retention curves — fit versus no fit (hypothetical cohorts). Hypothetical illustration. A flattening curve means a core group finds lasting value; decay to near zero means the product is a trial, not a habit.

1.11 Customer value versus business value

- **Customer value** — benefit the customer gets relative to what it costs them (money, time, effort, risk).
- **Business value** — what the organisation gains: revenue, margin, retention, strategic position, cost savings, learning.

Good products create value for customers *first*, then capture a fair share. Both must be positive or the product is not sustainable.

Situation	Customer value	Business value	Verdict
Clear fee display at checkout	High (trust, fewer surprises)	Medium-high (fewer abandons)	Win-win
Hidden fees revealed at the last step	Negative	Short-term gain, long-term trust loss	Reject — exploits users
Deep discount to win users	High for now	Negative contribution margin	Only as a bounded experiment
Heavy personalisation on sensitive data	Possible	Possible	Gate by consent, privacy and regulation

1.12 Output versus outcome

- **Output** — what the team produces: features shipped, tickets closed, story points.
- **Outcome** — the change in user behaviour or business results that follows: higher checkout completion, fewer support contacts, better retention.

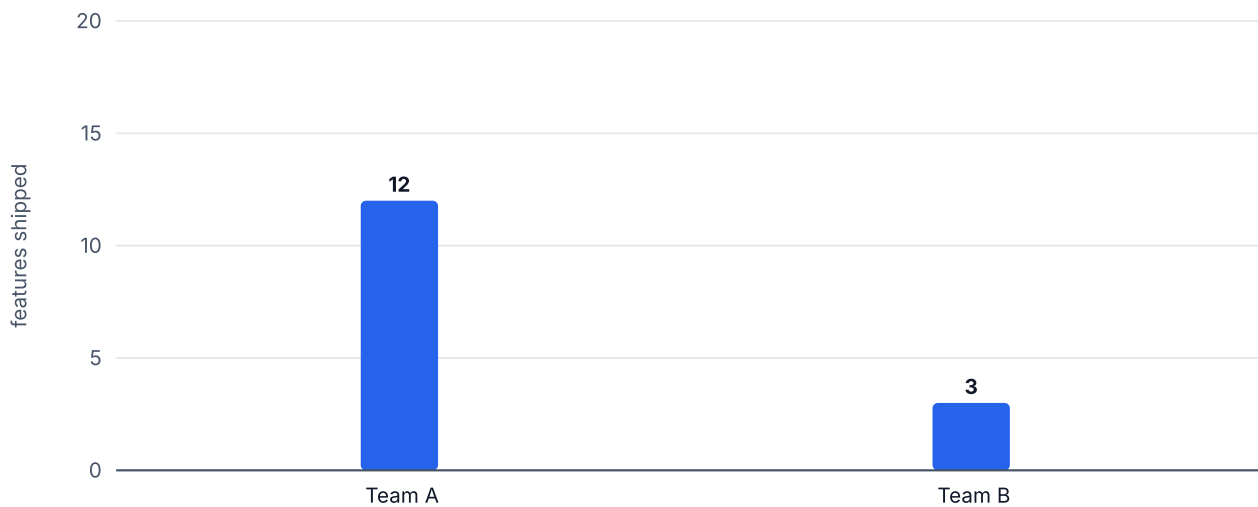
From your course notes (Companion ch. 1, Extra topic 2)

“A feature is something the team builds; an outcome is the change in user behaviour or business results. Neither a polished screen nor a shipped story alone proves value.” In OKR terms: if the release happens but completion does not improve, **the initiative shipped but the key result has not been achieved.**

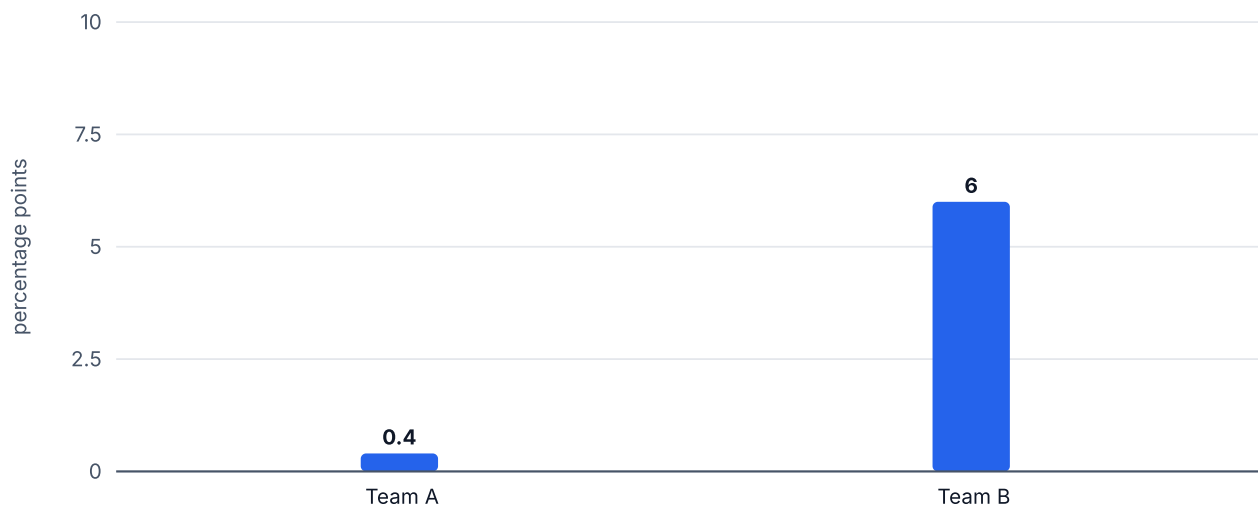
Hypothetical worked example · feature factory vs outcome focus

A food-ordering team ships 12 features in a quarter. Checkout completion is **70.0%** before and **70.4%** after. Another team ships 3 changes: show the delivery fee early, add a retry for failed payments, remove one form field; completion rises from 70.0% to 76.0% and complaints do not increase.

Output: Team A 12, Team B 3. Outcome: Team A +0.4 pts, Team B +6.0 pts. The relevant question is not “how much did we build?” but “what changed for users and the business?”



Output (hypothetical quarter). Team A shipped four times as many features.



Outcome (same hypothetical quarter). Team B moved the metric 15× more with a quarter of the output. Output and outcome answer different questions.

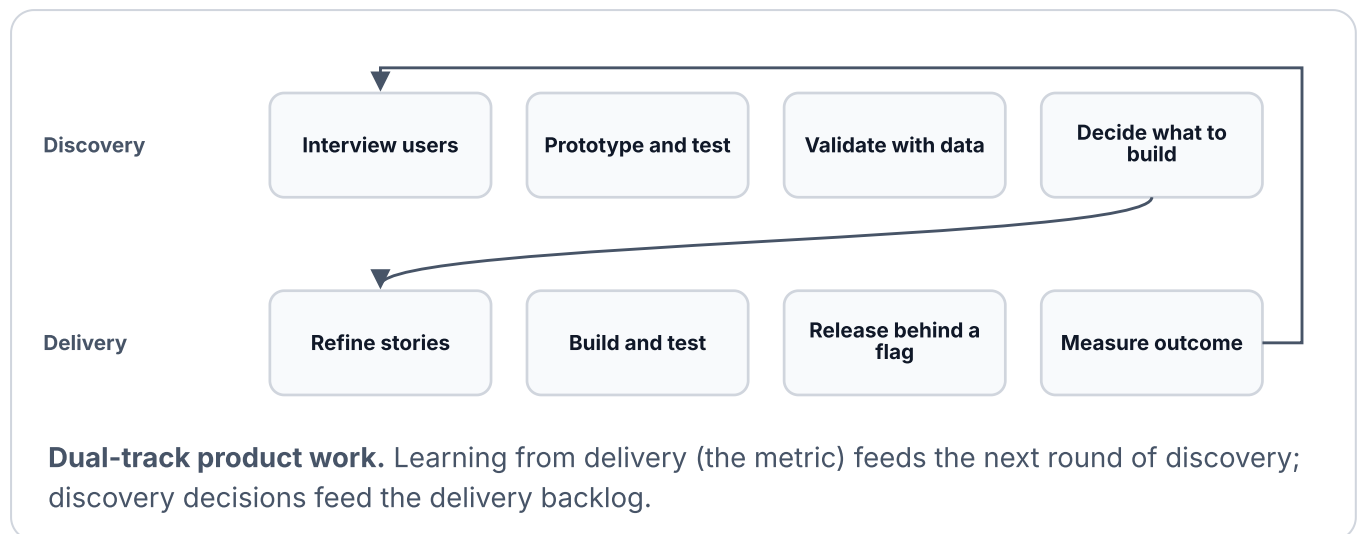
Level 3 · Why leaders care

Outputs are cheap to count and easy to inflate; outcomes are what pay for the organisation. Senior PMs translate roadmaps into **outcome language** for executives: "reduce failed payments by 30% this quarter," not "build payment retry screen." They also protect against Goodhart's law — when a metric becomes a target it stops being a good metric — by pairing each outcome with guardrails.

1.13 Product discovery versus product delivery

- **Discovery** — deciding *what* to build: understanding the problem, testing value and usability, estimating feasibility and viability.
- **Delivery** — building, testing and releasing it reliably.

In **dual-track** teams both happen continuously: discovery de-risks the next items while delivery builds the current ones.



When not to over-discover

If a change is small, reversible and cheap (a label, a default), just ship it behind a flag and measure. Reserve heavy discovery for costly, hard-to-reverse bets.

1.14 Stakeholder management (introduction)

A **stakeholder** is anyone affected by or able to affect the product: leadership, sales, support, legal, compliance, finance, operations, partners, and users.

Four habits cover most of it:

1. **Map them** — who has *influence* and who has *interest*? (Module 6 covers the power-interest grid.)
2. **Align on the goal first** — agree the outcome before debating solutions.
3. **Make trade-offs visible** — one page: options, cost, benefit, risk, recommendation.
4. **Close the loop** — record the decision, the owner and a review date.

From your course notes (Extra topic 11)

A large customer wants an export this week while the team is fixing payment failures. Compare user impact, revenue risk, urgency, engineering effort and operational risk; present a time-limited workaround or staged plan; record who decided and when to revisit.

1.15 Product sense and product thinking

Product sense is the ability to quickly see a product through the user's eyes and the business's, spot what matters, and propose sensible improvements. **Product thinking** is the habit of starting from the user problem and outcome rather than from a feature or technology.

Build it deliberately:

- **Critique products you use**, writing: who is it for, what job, what works, what frustrates, what would you test, what could go wrong.
- **Watch real people** use something — your own app or someone else's.
- **Read teardowns and case studies**, then predict the metric they would have moved.
- **Practise estimation** and metric reasoning (Modules 3 and 7).

From your course notes (Lectures 4, 19)

A critique should explain **the user, the task, the difficulty and the expected benefit** — not “the app is bad.” Separate **observations from assumptions**; an outsider can't know a company's internal data or reasons.

Worked example · a 90-second product critique (illustrative)

Product: a grocery-delivery app. **User:** a busy parent ordering weekly staples. **Job:** “Restock the kitchen without a store trip.” **Works:** saved baskets make repeat orders fast. **Pain:** delivery fee and minimum basket value are discovered late. **Hypothesis:** showing the fee and “add ₹X to avoid fee” in the cart reduces abandonment without hurting order value. **Metric:** checkout completion; **guardrail:** complaints and refund rate. **Risk:** the fee varies by distance — show an honest estimate or say “unavailable”, never an invented number.

1.16 Growth model and customer relationship: PLG versus SLG

	Product-led growth (PLG)	Sales-led growth (SLG)
Main acquisition engine	The product itself: self-serve sign-up, free tier or trial, virality	Sales team: outbound, demos, negotiation
Typical buyer experience	Try → get value → upgrade	Talk to sales → proof of concept → contract
Best for	Lower price points, easy individual adoption, bottom-up spread	High price/complex purchases, long procurement, heavy integration
PM focus	Activation, time-to-value, in-product conversion	Enterprise features, security/compliance, integrations, sales enablement
Key metrics	Activation rate, free→paid conversion, expansion	Win rate, sales cycle, ACV, net revenue retention
Risk	Big deals need sales-assist	Product neglected in favour of custom asks

Most mature B2B companies blend the two ("product-led, sales-assisted"). Module 8 develops growth loops and PLG in detail.

1.17 B2B versus B2C product management

Dimension	B2C	B2B
Buyer vs user	Usually the same person (sometimes different — e.g., a parent buys, a child uses)	Often different: buyer, admin, end-user, and approver
Volume	Large, fast feedback and statistical A/B tests	Fewer customers; qualitative depth matters more
Success metrics	Activation, retention, engagement, ARPU	Adoption within accounts, renewal, expansion, NRR
Sales cycle	Minutes to days	Weeks to months
Risk of customisation	Low	High — one large customer can steer the roadmap
Compliance	Privacy, consumer protection	Security reviews, SLAs, contracts, data residency

From your course notes (Lecture 20)

For B2B, the **buyer and the actual end-user may be different people** — test with both.

1.18 Platform products versus feature products

- A **feature product** solves a bounded job for end-users (a scheduling app).
- A **platform** provides capabilities others build on (payments API, identity service, internal developer platform). Its customers are often *developers or other teams*.

Platform PMs optimise **developer experience** (clear docs, fast integration, stable APIs, predictable pricing), **reliability** and **ecosystem growth**, and must manage **backwards compatibility** — breaking an API breaks someone else's product.

Platform trap

Building a platform before you have two or three real consumers usually produces abstractions nobody wants. Prove the use case with a feature first, then generalise.

1.19 Common Product Management mistakes

Mistake	Why it hurts	Fix
Starting with a solution	You optimise the wrong thing	Write the problem statement and evidence first
Confusing activity with progress	Shipping $\neq$ impact	Define the outcome metric before building
Treating stakeholder requests as requirements	Roadmap becomes a request queue	Ask for the underlying job; show trade-offs
Skipping discovery because "we know our users"	Confirmation bias	Do 5–8 interviews; watch behaviour
Over-relying on data (or on gut)	Blind spots either way	Combine quantitative "what" with qualitative "why"
Vague success metrics	Can't tell if you won	Define numerator, denominator, period, guardrails
Too many priorities	Nothing ships well	Limit WIP; make "not now" explicit
Weak relationships with engineering/design	Surprises, rework	Involve them early; ask for estimates and risks, don't dictate dates
No post-launch review	No learning	Schedule a review before launch
Inventing numbers	Loses credibility	Label assumptions; say "not measured" when true

Integrity note from your course

The guide repeatedly flags that candidates should **not misrepresent job titles, invent metrics or claim work they did not do**. Present course or personal projects as such, label assumptions, and say plainly when you did not measure a result. This is both ethical and practical: interviewers probe.

1.20 Source-flagged misconceptions (corrected)

- A **sprint** is ≤ 1 month — not necessarily two weeks.

- **MVP ≠ prototype ≠ public launch.** An MVP is the smallest useful version or *experiment* that teaches you something important; it must still be safe and fit its test.
- **Kanban is flow management with WIP limits**, not a Scrum–XP hybrid.
- **A focus group is a moderated research discussion**, not another name for a target audience.
- **There is no universal rule** that every product has two or three personas.
- A **unicorn** is a private company valued at $\geq$ US\$1 billion, not one with \$1 billion revenue.
- **Product–market fit is not proven** by market size alone.

1.21 Practice and model answers

Exercise 1 · Role clarity

Your company has a Product Owner, a Project Manager and you (PM). A VP asks for a new report “by end of next month.” Write who should decide (a) whether the report is worth building, (b) the scope for the first release, (c) the delivery plan and dates.

Model answer

(a) **PM** with the VP: this is a value/priority decision tied to outcomes. (b) **PM/Product Owner** with design and engineering: scope is a trade-off between value and effort. (c) **Project Manager** with engineering: schedule, dependencies, risks. The PM stays involved by monitoring scope trade-offs. State the assumption that these decision rights are typical, and confirm them with the team.

Exercise 2 · Turn an output into an outcome

“Build a chatbot for support” is on the roadmap. Rewrite it as an outcome with a metric, a baseline and a guardrail.

Model answer

Outcome: “Resolve common order-status questions without agent contact.” Metric: *self-service resolution rate* for order-status queries, baseline 35% (assumption — must be measured), target 55% in one quarter. Guardrails: customer satisfaction of bot-handled chats $\geq$ baseline; escalation-to-human success; incorrect-answer rate from sampled review.

Exercise 3 · Critique in 90 seconds

Pick an app you use daily. Using the structure *user* → *job* → *what works* → *pain* → *hypothesis* → *metric* → *guardrail* → *risk*, write a critique in under 150 words.

Model answer outline

Any app works; assess your answer against the structure above and the Level-1 test: could a non-expert understand the user and the problem? Did you separate observation from assumption? Did you include a guardrail?

1.22 Interview perspective

"What does a Product Manager do?" — 60-second answer

"A PM decides which problems are worth solving for which users, and defines how we'll know the solution worked. Day to day I talk to users and look at data to understand the problem, work with design and engineering to shape and deliver a solution, and align stakeholders on trade-offs. I'm accountable for outcomes — for example, improving checkout completion — not just for shipping features. I try to make decisions with evidence, write them down, and review results afterwards."

"How is a PM different from a Project Manager or Product Owner?"

What the interviewer tests: role clarity and humility about variation. *Structure:* define each by core question; note overlap; give a team example. *Answer:* "A project manager owns delivery of a defined scope on a timeline; a PM owns the problem, the outcome and the priority. A Scrum Product Owner manages the Product Backlog and its value for a team; in some companies the PM does that too. In my last team I treated the roles as decision rights: I'd confirm who decides scope, who decides dates, and who decides priority." *Mistake to avoid:* claiming PMs "manage" engineers or that titles mean the same everywhere.

1.23 Key takeaways

- A product is **value + experience + business model**.
- A PM's job is to **choose and clarify the problem** and be accountable for the **outcome**, working through influence, not authority.
- Retire the **four risks** — value, usability, feasibility, viability — as cheaply as possible.

- **Output ≠ outcome.** Define the metric and guardrails before building.
- **Discovery and delivery** run together in good teams.
- Role names vary; **decision rights** are what matter.
- Be honest: label assumptions and don't invent results.

1.24 Check your understanding

CHECK YOUR UNDERSTANDING

A team ships 12 features but checkout completion barely moves. Which statement is most accurate?

- A. They did great work because output was high.
- B. They produced high output but low outcome.
- C. Checkout completion is not a valid metric.
- D. They should ship more features.

Why: Output counts what was built; outcome measures the change in user or business results. High output with flat outcome is the feature-factory pattern.

In Scrum, how long can a sprint be?

- A. Exactly two weeks
- B. At most one month
- C. At least one month
- D. Any length, including quarterly

Why: The Scrum Guide sets sprints at one month or less; two weeks is a common choice, not a rule.

Which is NOT strong evidence of product-market fit?

- A. Retention curves that flatten
- B. Rising organic referrals
- C. A very large estimated TAM
- D. Many users saying they'd be very disappointed to lose the product

Why: Market size says there could be demand; fit requires evidence that real customers keep using and valuing the product.

Which risk does a usability test mainly reduce?

- A. Value risk
- B. Usability risk
- C. Feasibility risk
- D. Viability risk

Why: Usability tests check whether users can figure out how to use the solution. Value, feasibility and viability need other evidence.

A sales lead asks for a PDF export. What should the PM do first?

- A. Add it to the sprint
- B. Refuse the request
- C. Ask what the customer does with it and what outcome they need
- D. Ask engineering for a date

Why: Requests are solutions; the underlying job determines the best option and the cost-benefit.

Which model most strongly encourages empowered teams?

- A. Feature factory
- B. Project-based delivery with fixed scope
- C. Teams assigned problems and outcomes
- D. Waterfall

Why: Empowered teams are given problems to solve and measurable outcomes, not a list of solutions.

1.25 Recommended resources

- [The Scrum Guide](#) — official rules for Scrum.
- [Atlassian: Product management](#) — practical overviews (from your course reference list).
- [Silicon Valley Product Group articles](#) — Marty Cagan on empowered teams and the four risks.
- [Product Talk \(Teresa Torres\)](#) — continuous discovery habits.
- [Lenny's Newsletter](#) — practitioner essays and PM career guidance.