

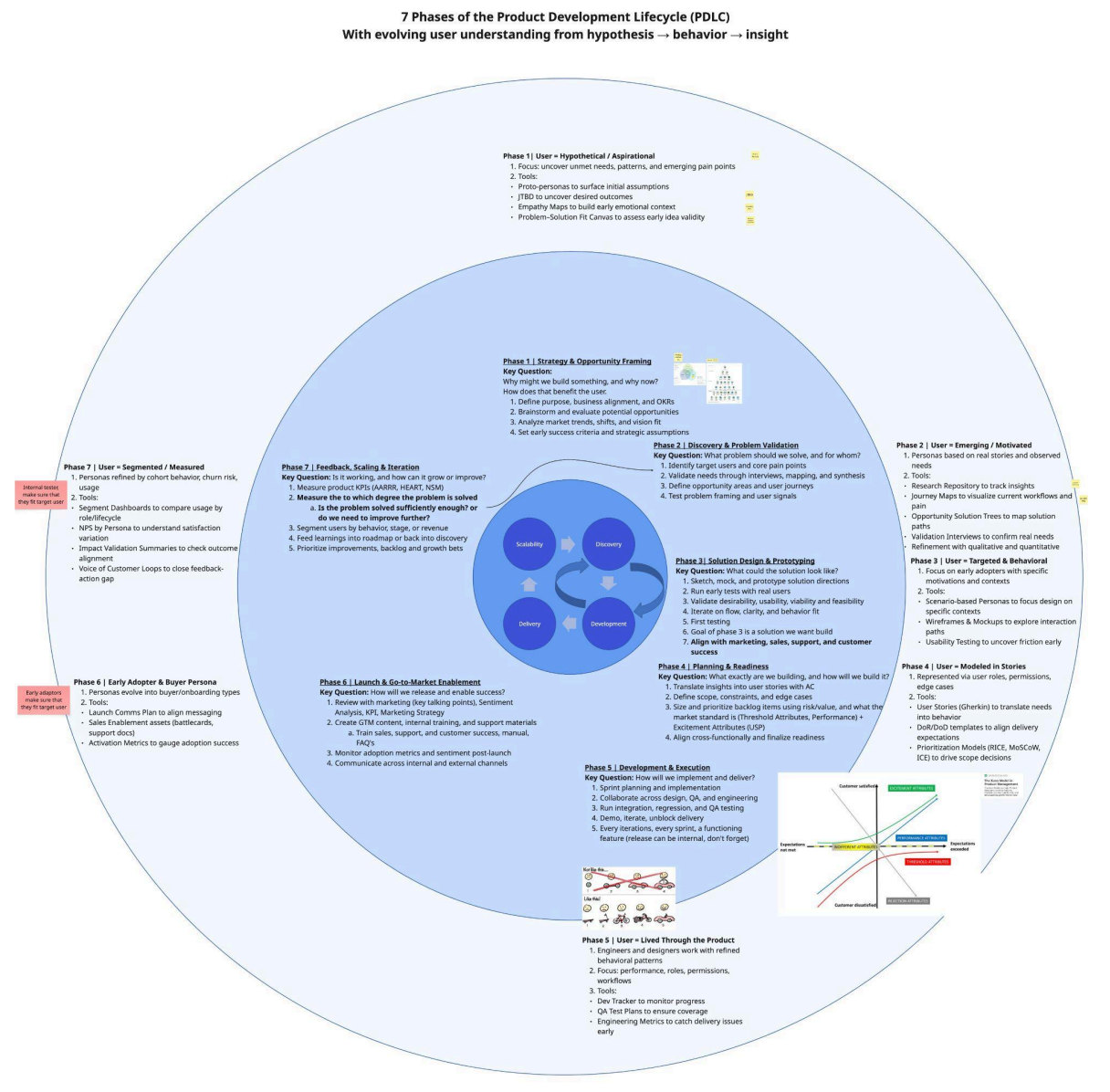
PM Playbook

Christof Gernhardt

7 Phases of the Product Development Lifecycle (PDLC).....	1
When to Use This Playbook.....	2
Situation.....	2
Go To Section.....	2
Strategy & Planning.....	2
Key Actions:.....	2
Frameworks & Artifacts:.....	3
Tips:.....	3
Discovery & Validation.....	3
Key Actions:.....	4
Frameworks & Artifacts:.....	4
Tips:.....	5
Requirements & Planning.....	5
Key Actions:.....	5
Frameworks & Artifacts:.....	5
Tips:.....	6
Validation & Testing.....	6
Key Actions:.....	6
Frameworks & Artifacts:.....	7
Tips:.....	7
Launch & Rollout.....	7
Key Actions:.....	8
Frameworks & Artifacts:.....	8
Tips:.....	8
Feedback & Optimization.....	9
Key Actions:.....	9
Frameworks & Artifacts:.....	9
Tips:.....	10
Team & Ways of Working.....	10
Key Actions:.....	10
Frameworks & Artifacts:.....	10
Tips:.....	11
Compliance & Security.....	11
Key Actions:.....	11
Frameworks & Artifacts:.....	11
Tips:.....	12

7 Phases of the Product Development Lifecycle (PDLC)

With evolving user understanding from hypothesis → behavior → insight




When to Use This Playbook

Situation	Go To Section
Need product direction or vision clarity	1. Strategy & Planning
Want to understand what users need	2. Discovery & Validation
Have ideas, but unsure how to define them clearly	3. Requirements & Planning
Ready to test a new feature or MVP	4. Validation & Testing
About to launch something	5. Launch & Rollout
Need to improve or learn from feedback	6. Feedback & Optimization
Team process isn't working or feels off	7. Team & Ways of Working
Working with sensitive data or legal concerns	8. Compliance & Security

Strategy & Planning

Purpose: Set the long-term direction, business context, and success measures for the product.

 *Start here:* Use the Lean Canvas and North Star Metric to shape your product's direction.

Key Actions:

- Define or help define depending on the company and maturity of the company, and communicate the product vision, mission, and principles
- Establish OKRs (SMART) and a North Star Metric to align and measure success
- Build and maintain a Now–Next–Later product roadmap
- Analyze the market using TAM/SAM/SOM, Porter's Five Forces, Lean Canvas
- Segment users and evaluate willingness to pay
- Identify user personas and define their Jobs To Be Done (JTBD)
- Define the revenue model and clarify the market category
- Map and analyze stakeholders using Power/Interest and create a stakeholder register
- Document strategic themes and measurable success criteria
- Validate product–market fit using PMF surveys and early traction indicators
- Analyze the competitive landscape to define positioning and differentiation
- Document and revisit strategic assumptions that underpin your product bets
- Set a regular cadence for vision alignment reviews (e.g., quarterly with OKR review)

Frameworks & Artifacts:

- Vision, Mission, Product Principles
- Now–Next–Later Roadmap
- OKRs, North Star Metric
- Lean Canvas, TAM/SAM/SOM, Porter’s Five Forces
- Target Personas, JTBD Framework
- Market Fit Analysis, Willingness to Pay Models
- Product–Market Fit Survey Results, Traction Metrics
- Competitive Landscape Map (e.g., SWOT, 2x2 positioning)
- Strategic Themes or Pillars
- Vision Alignment Review Schedule
- Stakeholder Map (Power/Interest Grid), Stakeholder Register
- Strategic Assumptions Log

Tips:

- Anchor strategy in problems, not ideas. Vision should solve real user pain.
- Revisit your North Star Metric in every roadmap and OKR review—it aligns execution to purpose.
- Use product principles as filters for what not to build.
- Validate your PMF assumptions continuously; they are not static.
- Involve key stakeholders early—surprises in strategy often come from missed alignment.
- Make vision reviews a quarterly habit, just like retros or sprint reviews.

Discovery & Validation

Purpose: Understand user problems, validate opportunities, and explore solutions to reduce risk before development.

 *Start here:* Interview 3 users. Fill a Discovery Table. Identify 1 key assumption to test.

Key Actions:

- Interview users and gather feedback through structured qualitative research
- Run surveys and analyze usage data to validate insights (triangulate evidence)
- Create and refine problem and opportunity statements
- Frame and prioritize assumptions and hypotheses
- Plan and execute experiments (prototypes, MVPs, A/B tests) using learning cards
- Create and analyze user flows, task flows, and wire flows to visualize user behavior
- Define product solutions using the Why–How–What model and Solution Definition Canvas

- Facilitate ideation sessions with techniques like “How Might We...”, Brainwriting, and Crazy 8s
- Prioritize opportunities using multiple frameworks (RICE, Effort–Impact, MoSCoW, etc.)
- Choose the right MVP (Minimum Viable Product) or PoC (Proof of Concept) type based on what needs validating (value vs. feasibility)
- Define and run moderated/unmoderated user tests with clear goals and metrics
- Connect discovery outputs to strategic goals via a Vision-to-Experiment Tree
- Track insight signal strength and link findings to specific user personas and journeys

Frameworks & Artifacts:

- Discovery Table (with Signal Strength column)
- Value Proposition Canvas & Solution Definition Canvas
- User Personas, User Journey Maps
- JTBD Framework: “When [situation], I want to [motivation], so I can [outcome]”
- Research Repository & Interview Snapshot Template
- Problem Statements, Opportunity Areas
- Opportunity Solution Trees, Assumption Mapping
- Research Logistics Hexagon
- Ideation Setup: Passive vs. Assertive, “How Might We”
- Ideation Methods: Brainstorming, Brainwriting, Mind Mapping, Crazy 8s
- Prioritization Models: RICE, MoSCoW, Effort–Impact, Dot Voting, Scorecards
- MVP & PoC Strategy
 - When to Use / Not Use a PoC
 - MVP Types: Wizard of Oz, Concierge, Landing Page, Email, Crowdfunding, Piecemeal, Single-feature
- Prototyping Strategy: Types, Goals, Metrics, Improvement Plan, Design Sprint
- User Flows, Task Flows, Wire Flows
- User Testing Plan (method, goals, success criteria)
- Learning Cards / Experiment Tracker
- Working Backwards Press Release
- Vision-to-Experiment Tree
- Desirability–Feasibility–Viability Triage Grid
- Persona → Journey → Feature Mapping Grid

Tips:

- Don’t just validate ideas, validate that the problem exists and matters.
- Use triangulation: insights confirmed by interviews and data are high-confidence.
- Record user interviews consistently, use templates and tag recurring themes.
- MVPs validate value, PoCs validate feasibility don’t confuse them.
- Add signal strength to every finding to separate noise from action.
- Discovery doesn’t end at delivery and embed continuous discovery habits in your team.

Requirements & Planning

Purpose: Define what to build, for whom, and under what constraints by translating validated discovery into clearly scoped, prioritized, and testable product requirements.

 *Start here:* Draft one user story using Gherkin format. Review it with design or engineering.

Key Actions:

- Translate validated discovery into actionable requirements
- Write user stories with clear, testable acceptance criteria (Gherkin or similar)
- Break work into small, prioritized increments with edge cases and DoR (Definition of Ready) /DoD (Definition of Done)
- Run regular backlog refinement sessions with input from design, engineering, and QA
- Capture both functional and non-functional requirements (performance, accessibility, etc.)
- Document known constraints (technical, legal, compliance, UX)
- Identify and manage risks (viability, feasibility, usability, scope, dependencies)
Create and maintain a Dependency Map for system or stakeholder integrations
- Conduct strategic prioritization: What to build and why (across teams and leadership)
- Conduct tactical prioritization: How to build, sequence, and unblock (with delivery teams)
- Use the Vision-to-Experiment Tree to maintain alignment from problem → hypothesis → delivery
- Maintain traceability between persona → journey → opportunity → feature → story

Frameworks & Artifacts:

- Feature Specifications
- User Stories with Acceptance Criteria (Gherkin or Example Mapping)
- Requirement Statements (Functional & Technical)
- Definition of Ready (DoR) / Definition of Done (DoD)
- Edge Case & Risk Logs (Business Viability, Publicity, Value, Usability, Feasibility, Dependencies, Scope)
- Non-Functional Requirements (NFR) Checklist
- Constraint Log (Technical, Legal, Compliance, UX)
- Prioritization Models:
 - RICE
 - MoSCoW
 - Cost of Delay (CoD)
 - ICE
 - Value vs. Effort Matrix
 - Buy-a-Feature Exercise
- Dependency Map or Integration Checklist
- Refinement Guidelines

- Vision-to-Experiment Tree
- Opportunity Solution Tree
- Product Stack Overview
- Persona → Journey → Feature Mapping Grid

Tips:

- Write user stories from the user's perspective, not the implementation layer.
- Add edge cases and failure conditions into acceptance criteria up front.
- Treat non-functional requirements (e.g., latency, accessibility) as first-class citizens.
- Keep backlog refinement lean, focused, and recurring. Don't over-document.
- Use visual models like dependency maps and prioritization matrices to align with stakeholders quickly.
- A bloated backlog is a sign of indecision. Keep it clean and confidence-based.

Validation & Testing

Purpose: Ensure features function correctly, solve the intended user problems, and meet quality and usability expectations before launch.

 *Start here:* Build a basic usability test (5 tasks, 5 users). Focus on where people get stuck.

Key Actions:

- Define test coverage for functional, UX, edge case, and non-functional requirements
- Execute sprint-level validation: functional, regression, exploratory testing
- Facilitate User Acceptance Testing (UAT) with scripts and stakeholder sign-off
- Run moderated and unmoderated usability tests with real users
- Analyze usability findings and update backlog or acceptance criteria
- Validate MVPs and UX flows against hypotheses and business goals
- Ensure product analytics are properly instrumented for post-launch validation
- Run accessibility tests (WCAG compliance, keyboard nav, screen reader support)
- Document and triage known limitations, issues, and trade-offs
- Capture insights using structured formats (e.g. Learning Cards)
- Use bug severity matrices to prioritize test outcomes
- Update backlog items and roadmap priorities based on test results

Frameworks & Artifacts:

- Sprint Testing Plan (Functional, Regression, Exploratory)
- Exploratory Testing Charter
- Test Case Management System
- User Acceptance Testing (UAT) Scripts and Sign-off Tracker
- Usability Testing Plan (goals, tasks, scenarios, participants, metrics)


- Usability Testing Results & Reports
- UX Review Checklist (incl. heuristics, responsiveness, interaction patterns)
- Pre-release Launch Readiness Checklist
- Known Limitations / Trade-Offs Log
- Acceptance Test-Driven Development (ATDD) artifacts
- Prototype Evaluation Tools:
 - Heatmapping
 - A/B Testing
 - Sequential A/B Testing
 - Multivariate Testing
- Accessibility Checklist (WCAG 2.1, screen reader, keyboard support)
- Learning Cards / Insight Summaries
- Bug Severity Matrix / Triage Criteria

Tips:

- Validate the right thing (problem-solution fit) before validating the thing right (implementation quality).
- Don't wait until development ends! Test early with prototypes and flows.
- Use exploratory testing charters to give structure without restricting creativity.
- Document insights, not just issues. Done failed tests teach you more than passed ones.
- Treat usability bugs with as much weight as functional ones, as they cost trust.
- Accessibility is not optional. Make it part of every UX and dev acceptance gate.

Launch & Rollout

Purpose: Coordinate internal and external readiness, ensure smooth deployment, and set the foundation for a successful and recoverable release.

 *Start here:* Create a mini launch checklist (docs ready, support briefed, metrics defined).

Key Actions:

- Finalize the Go-to-Market (GTM) plan with clear goals, timing, and owner alignment
- Align all stakeholders: product, engineering, marketing, sales, support, legal, compliance
- Train internal teams (sales, support, field, execs) using enablement materials and live demos
- Create and distribute launch materials: FAQs, battlecards, help docs, blog posts, press kits
- Execute internal dry runs or pilot launches (canary, flags, soft launch)

- Confirm technical readiness: environments, instrumentation, feature flags, rollback plans
- Coordinate communications across all channels (internal + external) with a messaging plan
- Monitor post-launch metrics and usage in real time (dashboard, analytics, alerts)
- Establish support readiness and incident response plans for Day 0
- Conduct a launch retrospective and log all learnings for the next cycle

Frameworks & Artifacts:


- Go-to-Market Plan
- Launch Timeline / Event Calendar
- Internal Enablement Assets:
 - Battlecards
 - Demo Scripts
 - Internal FAQs
 - Sales & Support Briefs
- Stakeholder Readiness Docs & Sign-offs
- Launch Checklist (comms, content, support, engineering)
- Launch Communications Plan (audiences, messages, timing, channels)
- Demo Plan & Schedule
- Rollout Strategy Document:
 - Canary Release
 - Phased Rollout
 - Feature Flags
- Post-Launch Monitoring Plan (KPIs, health checks, instrumentation)
- Incident Response Plan / Escalation Paths
- Launch Retrospective Template / Lessons Learned Log

Tips:

- A good launch isn't one with no problems. It is one where you're prepared when they happen.
- Use internal launches to pressure-test your messaging, support flows, and analytics before going live.
- Don't just celebrate the launch, celebrate the learning. Make launch retrospectives part of your culture.
- Track enablement adoption (e.g., sales demo completion, internal FAQs used) to catch readiness gaps early.

Feedback & Optimization

Purpose: Measure product performance, gather actionable insights, validate impact, and fuel continuous iteration.

 *Start here:* Read recent NPS/support tickets. Pick one insight and trace it back to a feature.

Key Actions:

- Monitor product KPIs (AARRR, HEART, North Star Metric) and user behavior
- Collect and analyze qualitative feedback from support, NPS, surveys, and social channels
- Tag and theme feedback to identify high-impact trends
- Conduct retrospectives after major launches or iterations
- Evaluate whether features delivered their intended impact (impact validation)
- Run internal dogfooding loops to uncover usability or reliability issues
- Conduct postmortems after incidents or major regressions
- Feed validated insights into discovery, backlog, and roadmap prioritization
- Reassess alignment to North Star Metric post-release
- Share structured feedback summaries with product, design, engineering, and GTM teams

Frameworks & Artifacts:

- Product Metrics Dashboards: AARRR, HEART, North Star Metric
- Feedback Channels & Tools: NPS, CSAT, support systems, user interviews, social listening
- Feedback Repository / Tagging System (with themes, sources, severity)
- Impact Validation Summary (Expected vs. Actual Outcomes)
- Feedback Prioritization Grid (e.g., Effort vs. Impact or Severity vs. Frequency)
- Retrospective Templates:
 - Start/Stop/Continue
 - 4Ls (Liked, Learned, Lacked, Longed for)
- Incident Postmortem Template
- Dogfooding Tracker / Log
- Feedback-to-Backlog Pipeline
- North Star Alignment Review

Tips:

- Don't chase every piece of feedback → look for patterns with impact.
- A feature's success isn't that it shipped. It's that it created measurable value.
- Tag feedback by source and persona type to avoid overreacting to outliers.
- Dogfooding helps, but real users > internal assumptions.
- Feed learnings back into discovery. The best insights often come after launch.

- Use retros for learning, not blame. Run them even when launches go well.

Team & Ways of Working

Purpose: Define team roles, collaboration norms, and delivery rhythms to foster healthy, high-performing product teams.



Start here: Join all team rituals for 2 sprints. Take notes. Suggest 1 improvement.

Key Actions:

- Define roles, responsibilities, and ownership boundaries across PM/PO, design, engineering, QA, and data
- Document escalation paths for blockers, decisions, and stakeholder misalignment
- Establish and run agile ceremonies: planning, standups, reviews, retrospectives, refinements
- Set and maintain an effective meeting cadence across product and delivery
- Track team health and delivery metrics (velocity, predictability, quality)
- Run periodic team health check-ins (monthly or quarterly pulse)
- Facilitate knowledge sharing: demos, design reviews, internal tech talks, brown bags
- Define async collaboration guidelines and documentation expectations (esp. for distributed teams)
- Review meeting rituals quarterly for effectiveness and fatigue

Frameworks & Artifacts:


- Roles & Responsibilities Matrix (e.g., RACI or DACI)
- Contact Directory
- Escalation Paths
- Team Charter / Working Agreement (conflict resolution, decision-making, communication)
- Agile Process Overview (Scrum, Kanban, or hybrid)
- Refinement Guidelines / Definition of Ready Template
- Tool Stack Inventory:
 - Jira, Confluence, Slack, Figma, Miro, Notion, Loom, etc.
- Meeting Cadence Calendar
 - Planning, Standups, Reviews, Retrospectives, Refinement
- Team Health Radar (e.g., Spotify model or pulse survey tool)
- Knowledge Sharing Rituals Calendar (demo days, show & tells, brown bags)
- Delivery Dashboards (velocity, sprint completion rate, blocker trends)

Tips:

- A healthy team is more valuable than any one feature. Invest in relationships and rituals.
- Meetings should have a purpose, owner, and timebox or they shouldn't exist.
- Culture is how decisions are made when you're not in the room.
- Distributed teams need clarity, not more meetings. Document decisions and default to async.
- Use retrospectives not just for sprint improvements, but to surface team health issues and morale dips.

Compliance & Security

Purpose: Ensure the product meets applicable legal, privacy, regulatory, and security standards throughout the lifecycle, without compromising user trust or product velocity.

 *Start here:* Ask your engineering lead which frameworks apply (e.g. GDPR, SOC2). Log what data is being stored and where.

Key Actions:

- Identify applicable compliance frameworks (e.g., GDPR)
- Engage legal, and data privacy early in product planning and refinement
- Document compliance and data handling requirements for new features
- Perform security threat modeling and risk assessments during planning
- Follow secure development practices and maintain a Secure Dev Checklist
- Ensure privacy by design: data minimization, user control, transparency
- Include legal and compliance sign-off in pre-launch checklists
- Maintain audit logs of product changes related to sensitive data or systems
- Coordinate incident response plans with cross-functional teams
- Stay updated on regulatory changes and emerging obligations (e.g., AI Act, Digital Markets Act)

Frameworks & Artifacts:

- Compliance Requirements Matrix (per market and feature)
- Secure Development Checklist
- Privacy by Design Guidelines
- Data Handling Documentation (e.g., encryption)
- Security Risk Register / Threat Model
- Legal & Infosec Review Checklist
- Launch Sign-Off Checklist (incl. compliance & security)
- Audit Trail Documentation (for sensitive features, permissions, data flows)

- Incident Response Plan & Escalation Path
- Compliance Stakeholder Map (legal, security, engineering leads)

Tips:

- Involve legal early, not to slow you down, but to help you go fast without breaking things.
- Security and compliance are continuous, not one-time checkboxes. Bake them into your rituals.
- Document why you made risk decisions. Future you and auditors will thank you.
- Build privacy features as product features: give users control and clarity.
- Don't wait until a breach or audit to create your security documentation. Do it proactively.