

## MVPCompanion

# From an Uncertain App Idea to a Testable MVP Specification

*A conceptual paper on progressive clarification, traceability, learning, and AI-supported product definition*

By Hans-Jörg Mathony · Version 1.0 · October 2026

**Abstract.** Early app ideas are often expressed as features, interfaces, or broad ambitions before the underlying problem, user, product logic, and evidence have been clarified. MVPCompanion addresses this uncertainty through a structured sequence of fifteen product questions. Its purpose is not to make uncertainty disappear, nor to guarantee product success, but to turn an initial idea into a coherent, testable, and buildable MVP specification. This paper explains the method's progressive logic, its five cross-cutting lenses, the Product Definition Chain, the principle of modelling the product before screens, and the role of the MVP as a learning instrument. It also explains how market and business questions, evidence discipline, traceability, and AI-supported work fit into the framework. AI is treated as a thinking partner—not as an autonomous product decision-maker or a source of evidence. The paper uses the documented MyBookScout case as an illustrative methodological reconstruction and distinguishes documented material from hypotheses, derived reasoning, decisions, results/status, and open questions.

## 1. Introduction: The challenge of transforming an uncertain app idea into a testable specification

An app idea often begins with an understandable but incomplete intuition: a group of people experiences friction, a familiar workflow could be improved, or existing products do not quite meet a need. The idea may quickly acquire a name, a feature list, or a sketch of the interface. Those artifacts can create a sense of progress, but they do not yet establish what problem matters, whose problem it is, what outcome would count as valuable, or what the smallest useful product must do.

The central difficulty is not simply a lack of detail. It is the presence of linked uncertainties. The problem may be misidentified; the target user may be too broad; the proposed value may not be distinctive; the product model may be confused with its screens; and the technical or commercial assumptions may be untested. If decisions are made independently, the resulting specification can be internally inconsistent: features may not serve a clear job, requirements may not support the intended outcome, and metrics may reward activity rather than user value.

MVPCompanion is designed to address this problem through progressive clarification. Each stage asks a specific product question and produces an output that can inform the next decision. The method's promise is practical rather than predictive: help a team move from an uncertain idea toward a coherent, buildable, and learnable MVP while making assumptions and unresolved questions visible.

This paper explains the method as a connected framework. It does not claim that following the process guarantees product-market fit, commercial success, or technical delivery. Nor does it treat the MyBookScout case as proof that its product hypotheses have been validated. The case is used to show how the method can organize product reasoning while preserving the difference between what is documented and what remains uncertain.

## 2. MVPCompanion as Progressive Clarification: The overall method and its logic

Progressive clarification is the central logic of MVPCompanion. Instead of attempting to specify an entire product in one leap, the method breaks the work into fifteen sequential clarification questions. Each stage narrows a different kind of uncertainty and produces a defined output. That output becomes an input to later reasoning, while later discoveries may expose a need to revisit earlier assumptions.

The canonical sequence is: S01 Idea; S02 Problem; S03 User; S04 Value; S05 Product; S06 Context & Lifecycle; S07 Core Jobs; S08 Scope; S09 Experience; S10 Information Architecture; S11 Data; S12 Requirements; S13 Learning; S14 Technical Boundaries; and S15 MVP Boundary. The sequence matters because downstream detail depends on upstream clarity. Requirements are difficult to make meaningful if the problem and intended outcome are vague; technical boundaries are difficult to assess if the product model and scope remain unstable.

The sequence should not be mistaken for a rigid one-pass waterfall. Clarification is progressive, but product work remains iterative. Evidence from a prototype, market investigation, or technical spike may require the team to revisit a previous decision. The stages provide a disciplined order for asking questions, not a claim that uncertainty will always decline smoothly or that every decision becomes final when its stage is complete.

The broader journey can be expressed as IDEA → OPPORTUNITY → PRODUCT → BUSINESS → MVP → SPECIFICATION → BUILD → LEARN. The fifteen stages provide the core clarification path within that journey. They are complemented by five cross-cutting lenses—Desirability, Differentiation, Viability, Feasibility, and Learning—which are applied throughout the process rather than treated as five additional stages. This distinction prevents the method from confusing the sequence of work with the perspectives used to assess it.

A useful working rhythm is: clarify the question, examine available information, distinguish evidence from assumptions, make a provisional decision, record the output, and identify what remains open. The method is successful at this point when the next decision is better framed—not when every uncertainty has been eliminated.

## 3. The Five Lenses: The perspectives used to examine the emerging product

The five lenses help prevent a product from being defined through only one perspective. A team can build something technically possible but unwanted, desirable but indistinguishable, useful but commercially unsustainable, or easy to describe but difficult to operate. Learning must also be considered explicitly: an MVP should create an opportunity to resolve important uncertainty.

Desirability asks whether people want the outcome the product promises. It directs attention to the problem, user context, motivation, and actual behavior rather than relying only on enthusiasm for an idea. Differentiation asks why a user would choose this product rather than an alternative, including existing habits, competing products, manual workarounds, or a general-purpose tool. Viability asks whether the product can create and capture value over time. This includes value exchange, customer and pricing hypotheses, acquisition assumptions, and the costs of delivery and operation.

Feasibility asks whether the intended product can be built and operated within relevant technical, data, organizational, and operational boundaries. Learning asks what the MVP needs to make clearer, which assumptions are most consequential, and what observable evidence would change the team's view. Learning is not simply a final testing phase; it is a lens that should shape the scope and measurement of the product from the outset.

The lenses are complementary, not interchangeable. Strong desirability does not establish viability; technical feasibility does not establish demand; and a differentiated concept does not prove that the difference matters to

users. A lens review should therefore expose tensions and trade-offs rather than produce a single unqualified score.

In the MyBookScout case, the proposed distinction is a shift from organizing reading data toward interpreting it for an individual reader. The intended value is better personal book decisions, especially the next-read decision. This is a differentiation and desirability hypothesis, not market validation. Viability depends on unresolved willingness-to-pay and business-model assumptions; feasibility depends on data quality, ranking, explanation, and operational choices; learning depends on whether readers find recommendations genuinely useful and continue to use the product.

## 4. The Product Definition Chain: Connecting the problem to measurable product requirements

A product specification becomes coherent when its elements are connected by explicit reasoning. MVPCompanion expresses this through a traceability chain: PROBLEM → USER → OUTCOME → JOB → CAPABILITY → REQUIREMENT → ACCEPTANCE CRITERION → METRIC. Each link answers a different question. The problem identifies the friction or unmet need; the user specifies who experiences it; the outcome describes the change that matters; the job describes what the user is trying to accomplish; the capability identifies what the product must enable; the requirement states observable system behavior; the acceptance criterion defines how that behavior will be judged; and the metric supports evaluation of the intended result.

The chain is not merely a documentation format. It is a test of reasoning. If a feature cannot be connected to a user job or intended outcome, its priority should be challenged. If a requirement has no acceptance criterion, it may not be testable. If a metric measures only clicks or feature usage while the stated outcome concerns decision quality, the measurement system may reward the wrong behavior.

The chain also exposes missing links. A broad problem statement may not identify a sufficiently specific user. A capability may be named without explaining which job it supports. An acceptance criterion may verify that a button works without verifying whether the underlying capability helps the user. These gaps should be resolved explicitly rather than hidden by more detailed prose.

In the MyBookScout case, the documented problem framing is the Reader's Paradox or Decision Fatigue: a reader has many books, reviews, recommendations, and TBR items but still struggles to decide which book is right for them now. The primary user is an individual reader with a library and a developed personal taste. The central job is to make a better next-read decision. The proposed product model connects Library, Taste, Discovery, Decision, and Learning. The core capability is a personalization loop that captures library signals, builds a taste understanding, recommends books, explains their fit, receives feedback, and improves.

The case materials identify useful-recommendation measures and related activation, save, start, read-conversion, feedback, and retention metrics. They also identify a Book DNA accuracy target above 70 percent after approximately twenty books. That figure is a target, not an achieved result. The chain is strongest when every requirement and metric can be traced back to the user problem and intended outcome, and when evidence gaps are clearly marked.

## 5. Product Model before Screens: Establishing product structure before detailed interface decisions

Screens are visible, concrete, and easy to discuss. For that reason, teams can begin designing them before agreeing on what the product fundamentally is. This creates a risk: the interface becomes the product definition, while the underlying objects, relationships, states, and rules remain implicit. MVPCompanion therefore places the Product stage and the Product Mental Model before detailed Experience and Information Architecture work.

A Product Mental Model describes the product’s meaningful concepts and how they relate. Depending on the product, these may include durable entities, user-owned objects, states, interactions, derived information, and the core loop through which value is created. A lifecycle model clarifies how an object or relationship changes over time. This structure provides a basis for deciding which user flows are needed, what information belongs where, and what data the system must preserve.

For a reading-intelligence product, the case materials distinguish bibliographic entities such as Work, Edition, ISBN, Author, and Publisher from user-specific entities such as UserBook, status, ownership, rating, notes, and preferences. Behavioral events include actions such as scanning, viewing, saving, skipping, starting, finishing, and rating. UserTaste is derived from signals, while recommendations have candidates, score components, evidence, and outcomes. These are conceptual model elements; the final relational schema, API contracts, and implementation details remain open.

The distinction between model and interface is not an argument against design. It makes design more purposeful. Once the product model and core jobs are understood, experience design can show how users interact with those concepts and how the product communicates its value. Information Architecture can then organize content and actions around the product’s functional logic instead of mirroring technical data structures or a generic feature list.

A product model is also a useful control against scope drift. If a proposed feature introduces a new object, state, workflow, or dependency, the team can ask whether it supports the core product promise or adds complexity without improving the primary outcome.

## 6. The MVP as a Learning Instrument: Connecting product scope to uncertainty and evaluation

An MVP is not simply a small version of a larger product, nor is it necessarily a rough or low-quality release. In MVPCompanion, the MVP is the smallest coherent product that can test the core promise and produce useful learning. Coherence matters: a collection of minimal features is not an effective MVP if those features do not form a usable path to the intended outcome.

Scope should therefore be derived from the central hypothesis and the minimum capability set needed to test it. The team should identify what must be present for the user to experience the proposed value, what can be simplified without invalidating the test, and what should remain outside the boundary. This is different from prioritizing a long feature list by convenience or stakeholder preference.

The MyBookScout case frames a central hypothesis: readers may be willing to provide their library if doing so produces visibly better and more relevant personal recommendations. The proposed core flow is library capture → taste understanding → recommendations → explanation → feedback. The documented scope hypothesis includes onboarding, scanning and manual book addition, library and status management, basic taste understanding, recommendations, explanations, feedback, and basic TBR prioritization. Social feeds, clubs, messaging, gamification, e-reading, marketplaces, and non-decision-useful statistics are listed as exclusions.

An MVP becomes a learning instrument only when its uncertainties are linked to observations and decision rules. Before testing, the team should specify what it expects to observe, which signals would support or challenge the hypothesis, and what it would do in response. Recommendation volume alone is not enough if the promise is useful decisions. Measures such as saves, starts, completed reading, feedback, perceived fit, and retention may provide complementary evidence, but each requires clear definitions and an appropriate test design.

The distinction between target and result is essential. A proposed threshold, planned beta, or illustrative metric is not evidence that the product works. The case materials describe a planned beta of 50–100 readers and

a set of validation approaches, but the reviewed materials do not establish actual test results. The method should preserve that open status instead of retrospectively treating the hypothesis as validated.

## 7. Traceability: Preserving the relationships between decisions and artifacts

Traceability preserves the reasoning that connects an initial problem to the final MVP boundary. It allows a team to answer not only what it intends to build, but why each element belongs in the product and how its contribution will be assessed. The Product Definition Chain is the principal conceptual path; the stage outputs provide the artifacts through which that path can be recorded.

Traceability must work in both directions. Forward traceability asks how a problem and user outcome lead to a job, capability, requirement, acceptance criterion, and metric. Backward traceability asks why a particular requirement exists, which job it supports, and which outcome justifies it. If a requirement cannot be traced backward, it may be unnecessary or insufficiently justified. If an important outcome has no downstream capability or metric, the specification may be incomplete.

The method also requires evidence-state discipline. The case matrix distinguishes Evidence, Hypothesis, Derived reasoning, Decision, Result/Status, and Open. Evidence is supported by an appropriate source, observation, or documented result. A Hypothesis is believed but not established. Derived reasoning follows from available information. A Decision records a choice made under current conditions. Result/Status records what happened or the current state. Open identifies unresolved questions or missing evidence.

These labels should not be collapsed. A decision can be sensible without being validated; a derived conclusion can be plausible without being directly observed; and an open question is not a defect to be concealed. In the MyBookScout case, the conceptual product model and proposed MVP boundary are documented, but user research, competitive validation, final technical choices, and actual MVP outcomes remain gaps in the available evidence.

Traceability is therefore a practical defense against retrospective certainty. The fifteen stages can be used to reconstruct and examine the product logic, but that does not prove the historical product development literally followed the sequence. Later knowledge must not be presented as if it had been available at the time of an earlier decision.

## 8. Cross-Cutting Workstreams: Integrating market discovery and business-model considerations

Market discovery and business-model reasoning are not isolated tasks that can be postponed until the product has been fully specified. They affect the interpretation of the problem, the definition of value, the scope of the MVP, and the conditions under which a product could be sustained. In MVPCompanion, business is a cross-cutting dimension of Viability rather than a separate generic business-plan exercise.

Market discovery examines the alternatives available to users, including direct competitors, adjacent products, existing habits, manual workarounds, and general-purpose tools. It should clarify what users do today, where those approaches fall short, and whether the proposed difference matters. A list of competitor features is not sufficient to establish differentiation. The relevant question is whether the product creates a meaningful advantage for a defined user and job.

Business-model work examines value exchange, customer hypotheses, pricing, acquisition, retention, cost to serve, and operational complexity. These assumptions can influence product choices. For example, a pricing

hypothesis may affect which capability is offered at which level; acquisition assumptions may make first-value time important; and data-provider costs or AI inference costs may alter the viable scope.

The MyBookScout case identifies a mature market across book tracking, cataloguing, social discovery, recommendations, analytics, and reading habits. It proposes a position around personal reading intelligence rather than feature parity. Its freemium model and indicative prices are explicitly hypotheses to test, not validated prices. Possible affiliate revenue is also constrained by a trust principle: commercial incentives should not distort recommendation ranking.

These examples show why viability should be examined continuously. Market and business information must be verified, and the product team should be explicit about the quality and recency of its sources. If the evidence is insufficient, the appropriate output is an open question or test plan—not a confident market claim.

## 9. Evidence and Reasoning: Making decisions defensible

A product specification is more defensible when readers can distinguish what is known, what is assumed, what follows from the current information, what has been decided, what has happened, and what remains unresolved. MVPCompanion’s evidence discipline makes those distinctions explicit. This matters because polished documents can make weak assumptions appear authoritative, especially when hypotheses are expressed in precise numbers or diagrams.

Evidence must be appropriate to the claim. A user interview may help explain a motivation but does not necessarily establish its prevalence. A prototype test may reveal usability problems but may not demonstrate retention or willingness to pay. Competitor documentation can establish what a product publicly offers, but it does not prove that users prefer one option. Metrics need clear definitions, suitable samples, and a test design that supports the intended inference.

AI output is not evidence by itself. AI can summarize provided material, suggest alternative interpretations, identify contradictions, or formulate questions for further investigation. Its output should be checked against the source material, and external, market, and business claims require verification. AI-generated citations, facts, book details, competitor descriptions, or numerical conclusions must not be accepted merely because they are fluent or plausible.

Reasoning should be inspectable. A useful record states the evidence considered, the assumptions that remain, the logic connecting them to a decision, and the conditions under which that decision should be revisited. This approach does not eliminate judgment; it makes judgment easier to challenge and improve.

The same discipline applies to visual models and tables. A figure should clarify a relationship or decision without implying more certainty than the underlying material supports. Color, layout, or emphasis must not turn an assumption into something that looks validated. Visuals should preserve the established labels and remain understandable without relying on color alone.

## 10. AI in MVPCompanion: Supporting work across the 15 stages

AI is a thinking partner within the MVPCompanion workflow, not the decision maker. Its role is to help users clarify, challenge, explore, structure, and review product reasoning. The human remains responsible for evaluating evidence, choosing among alternatives, making core product decisions, and accepting the resulting specification. This division is central to responsible use: AI can increase the breadth and speed of analysis, but it cannot confer validity on unsupported assumptions.

AI support is cross-cutting rather than confined to one stage. At S01 Idea, AI can help articulate an idea hypothesis and surface assumptions. At S02 Problem, it can challenge solution-first framing and propose questions that would distinguish a real problem from a presumed one. At S03 User, it can compare candidate

segments and identify what evidence would support prioritization. At S04 Value, it can help formulate value propositions and examine alternatives without treating differentiation as proven.

At S05 Product, AI can help describe the Product Mental Model and identify ambiguous concepts or relationships. At S06 Context & Lifecycle, it can enumerate states, transitions, and edge cases for human review. At S07 Core Jobs, it can help distinguish user jobs from proposed features. At S08 Scope, it can compare candidate capabilities against the core hypothesis and expose scope creep. At S09 Experience, it can challenge flows for missing states, friction, or delayed first value. At S10 Information Architecture, it can help organize information and actions around the product model and user jobs.

At S11 Data, AI can help identify entities, attributes, events, provenance questions, and data-quality risks, but it should not invent a final schema or external-data contract. At S12 Requirements, it can convert approved capabilities into draft observable requirements and acceptance criteria, then check their traceability. At S13 Learning, it can help formulate hypotheses, metrics, test plans, and possible interpretations while distinguishing targets from results. At S14 Technical Boundaries, it can help compare architectural options, dependencies, and failure modes; technical claims still require appropriate technical review. At S15 MVP Boundary, it can review the proposed in-scope and out-of-scope set for coherence and alignment with the central hypothesis.

A reliable AI prompt should define the input, the AI role, the task, the expected output, and the human review required. The prompt should also constrain the model to the supplied evidence and ask it to mark missing information rather than fill gaps with invention. For example, a review request can ask AI to map each requirement to a job and outcome, identify unsupported links, label every claim as Evidence, Hypothesis, Derived, Decision, Result/Status, or Open, and return a list of questions requiring human resolution.

AI can also help maintain continuity between stages by comparing successive artifacts for terminology drift, contradictions, missing outputs, and broken traceability. Such checks are especially useful when the specification grows. However, AI's review is itself fallible and must be checked against the canonical method and source artifacts. The human owns the final gate decision.

The MyBookScout technical concept illustrates a related but distinct product-architecture principle: structured systems rank; AI explains. In that proposed architecture, candidate retrieval, user-specific scoring, and constraint filtering are separated from explanation generation. AI explanations should be grounded in structured evidence and should not invent book facts or independently replace the ranking logic. This is a case-specific technical design principle, not a claim that every MVPCCompanion product must use the same architecture.

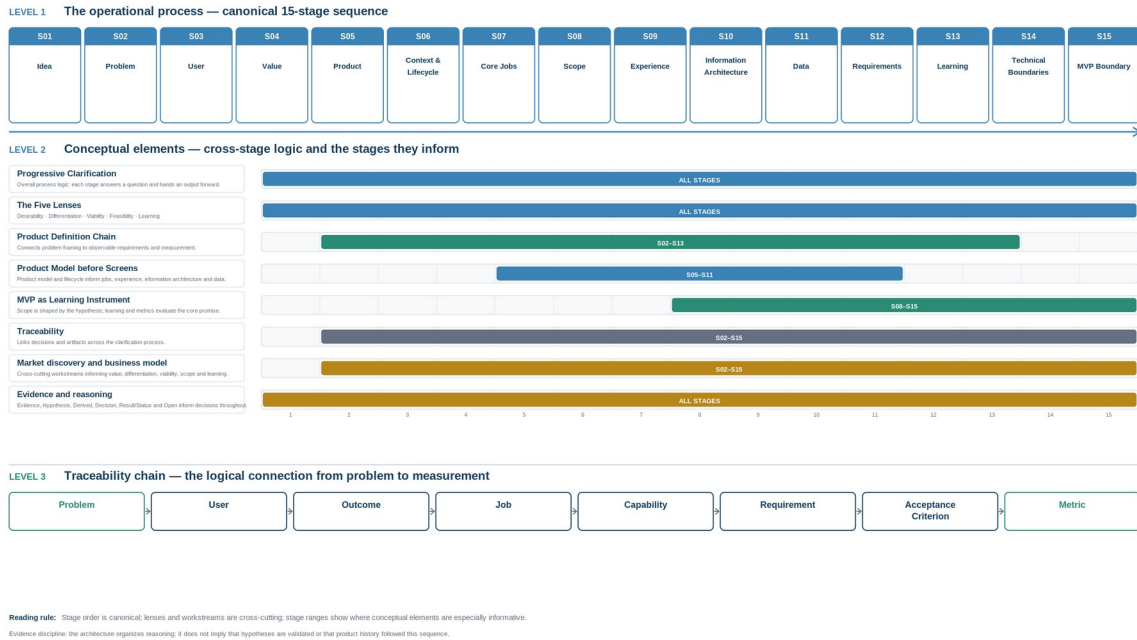
AI therefore has two distinct roles in this paper. First, it can support the human work of applying MVPCCompanion. Second, AI may be a component of the product being specified, as in MyBookScout's proposed explanation layer. These roles should not be confused: using AI to reason about a product does not imply that the product itself must be AI-based.

## 11. The Integrated Framework: Mapping the concepts and AI to the canonical 15-stage process

The framework becomes useful when its parts are treated as a connected system. The fifteen stages provide the sequence of clarification questions. The five lenses provide recurring perspectives for evaluating the emerging product. The Product Definition Chain preserves the reasoning from problem to metric. The Product Mental Model structures the product before detailed screens. The MVP boundary protects the smallest coherent test. Evidence discipline distinguishes what is known from what is assumed, and AI supports the work without taking ownership of the decisions.

## MVPCompanion's conceptual architecture

Three connected levels: operational process, conceptual elements, and traceability



**Figure 1. MVPCompanion's conceptual architecture: the operational process, cross-cutting conceptual elements, and Product Definition Chain.**

The figure connects three levels of the method. Level 1 shows the canonical sequence of 15 stages. Level 2 positions the eight conceptual elements in relation to the stages they inform, distinguishing process-wide principles from more specific relationships. Level 3 shows the Product Definition Chain, connecting the problem to the user, intended outcome, job, capability, requirement, acceptance criterion, and metric.

The architecture distinguishes the operational sequence from the concepts that inform it. The 15 stages provide the canonical order for progressively clarifying the product. The Five Lenses and the cross-cutting workstreams inform decisions across this sequence, while the Product Definition Chain and traceability connect decisions and artifacts. Evidence and reasoning support the decisions throughout, and AI can assist the work without replacing human judgment. The stage relationships shown in the figure are explanatory; they do not introduce additional stages or change the canonical process.

Table 1 summarizes the canonical stages, their principal outputs, and the kinds of AI assistance that may be useful. The AI column describes potential support, not mandatory automation or a new formal stage. Human review remains necessary, and the exact stage output is governed by the canonical specification.

| Stage       | Clarification question             | Principal output                | Possible AI support                                           |
|-------------|------------------------------------|---------------------------------|---------------------------------------------------------------|
| S01 Idea    | What could this be?                | Idea Hypothesis                 | Clarify the concept; surface assumptions.                     |
| S02 Problem | What real problem exists?          | Problem Statement               | Challenge solution-first framing; propose research questions. |
| S03 User    | Who has the problem?               | Primary Target User             | Compare candidate segments; identify missing evidence.        |
| S04 Value   | Why should this product matter?    | Value Proposition / Positioning | Draft and challenge value propositions and alternatives.      |
| S05 Product | What is the product fundamentally? | Product Mental Model            | Structure concepts and relationships; expose ambiguity.       |

|                              |                                       |                          |                                                        |
|------------------------------|---------------------------------------|--------------------------|--------------------------------------------------------|
| S06 Context & Lifecycle      | How does the product/object evolve?   | Lifecycle / States       | Enumerate states, transitions, and edge cases.         |
| S07 Core Jobs                | What must users accomplish?           | Prioritized Core Jobs    | Separate jobs and outcomes from features.              |
| S08 Scope                    | What is the smallest useful solution? | MVP Feature Set          | Test candidate scope against the core hypothesis.      |
| S09 Experience               | How does the user get value?          | Core User Flows          | Review flows, friction, and missing states.            |
| S10 Information Architecture | Where does everything belong?         | Information Architecture | Organize information around jobs and product logic.    |
| S11 Data                     | What information must exist?          | MVP Data Model           | Identify entities, events, provenance, and gaps.       |
| S12 Requirements             | What exactly must be built?           | Testable Requirements    | Draft observable requirements and acceptance criteria. |
| S13 Learning                 | How will we know it works?            | Metrics / Hypotheses     | Connect hypotheses to measures and test plans.         |
| S14 Technical Boundaries     | What must be technically decided?     | Technical Boundaries     | Compare options, dependencies, risks, and trade-offs.  |
| S15 MVP Boundary             | What is actually in and out?          | Final MVP Definition     | Check coherence, exclusions, and traceability.         |

*Table 1 — The canonical 15 stages connect each clarification question to a defined output; AI can support the work but does not replace human decisions.*

The integrated framework should be applied as a loop of disciplined handoffs. A stage output is reviewed before it becomes a basis for downstream decisions. The five lenses are applied across stages, not checked once and forgotten. Traceability is checked as requirements and metrics are created. Learning plans influence scope, and technical constraints may reveal a need to simplify or revise the product model. If evidence changes, the team can return to the relevant stage and update the linked artifacts.

The framework's reader-facing logic can be expressed as Understand → See → Do → Use AI → Check. Prose introduces the concept; a functional visual makes relationships visible; an exercise lets the reader apply the method; AI supports structured thinking; and a gate checks the output. This sequence is also a production principle for explaining MVPCompanion: the book should not only describe the method but enable readers to use it.

## 12. Conclusion: From an uncertain idea to a reasoned, testable, and buildable MVP specification

MVPCompanion offers a disciplined way to move from an uncertain app idea toward a coherent MVP specification. Its central contribution is not a promise of certainty but a structured way to ask better questions, make dependencies visible, and preserve the reasoning behind product decisions. The fifteen stages organize progressive clarification; the five lenses prevent one-sided product thinking; the Product Definition Chain connects user problems to requirements and metrics; and the Product Mental Model creates a foundation for experience, information architecture, and data decisions.

The MVP is treated as a learning instrument whose scope should be driven by a central hypothesis and a coherent path to user value. Traceability makes the specification inspectable, while evidence discipline prevents hypotheses, targets, and decisions from being mistaken for established results. Market and business considerations remain part of ongoing viability reasoning rather than being deferred to a separate planning exercise.

AI can strengthen this work by helping clarify, challenge, explore, structure, and review the artifacts produced across the stages. It can help detect missing links and contradictions, but it does not own product

decisions and its outputs do not constitute evidence by themselves. Where AI is also part of the product being specified, its responsibilities and boundaries should be explicit—for example, separating structured ranking from evidence-grounded explanation in the MyBookScout concept.

The method is most useful when it produces more than a polished document. It should leave the team with a defensible chain of reasoning, a visible account of uncertainty, a bounded product hypothesis, testable requirements, a plan for learning, and clear technical questions. That is the practical destination: not certainty before building, but enough coherent understanding to build deliberately and learn from what happens.

## Reference basis and evidence note

This paper is based primarily on the project’s MVPCompanion Book Production Specification, the canonical 15-stage specification, the MVPCompanion Method Specification and frozen method baseline, the Master Case Evidence Matrix v0.3, and the book’s visual/figure design guidance. The production baseline defines progressive clarification, the five lenses, the canonical stage order, the Product Definition Chain, AI Companion principles, and evidence-state terminology. The Master Case Evidence Matrix distinguishes documented MyBookScout material from hypotheses, decisions, results/status, and open evidence gaps.

The MyBookScout examples in this paper should be read as a methodological case analysis, not as proof of product success. In particular, the available case material does not establish validated user demand, confirmed pricing, completed validation experiments, a final technical stack, or a complete historical chronology connecting BookScout, MyBookScout, and Matchshelf. Such gaps are retained as open rather than filled by inference.