



SERIALLABS / PATTERN-TO-PRODUCT

PATTERN-TO-PRODUCT

How recurring operational problems can become reusable systems,
owned products and, under stricter conditions, new ventures

A PRACTICE-LED RESEARCH PAPER

Patterns become products. Products may become platforms. Platforms may create the conditions
for new companies. None of these transitions is automatic.

VERSION 1.3 / 27 AUGUST 2026

Prepared by SerialLabs

Contents

Executive summary.....	3
Abstract.....	4
Reader's note: epistemic status.....	4
1. The proposition.....	5
2. Intellectual foundations.....	8
3. Definitions and unit of analysis.....	11
4. The Pattern-to-Product model.....	14
5. Discovering a pattern.....	16
6. From pattern candidate to working system.....	18
7. From working system to reusable core.....	21
8. The productisation gate.....	24
9. Product, platform and venture.....	27
10. Governance, rights and responsible learning.....	29
11. The economics of Pattern-to-Product.....	31
12. Pattern-to-Product and Cognitive Workflow Systems.....	35
13. Testable propositions.....	36
14. Research programme.....	38
15. Operating model for SerialLabs.....	40
16. Illustrative pattern journeys.....	42
17. Failure modes and counterarguments.....	44
18. Managerial implications.....	46
19. Conclusion.....	47
20. Put the method to work.....	48
Appendix A. Pattern Candidate Record.....	49
Appendix B. Productisation Gate Record.....	50
Appendix C. Measurement catalogue.....	50
Glossary.....	52
References.....	52

Executive summary

Pattern-to-Product is the method SerialLabs uses to decide whether recurring operational work should become a bounded system, a reusable capability, an owned product or, under stricter conditions, a platform or venture. It begins inside the work rather than with a technology category. Repeated exceptions, fragmented evidence, manual coordination and delayed decisions are treated as signals to investigate - not as automatic product ideas.

The practical entry point is the **AI Opportunity Sprint**: one recurring process, one bounded decision and a 2-3 week diagnostic. It produces a current-state diagnosis, a Pattern Candidate Record, named evidence gaps and a recommendation to stop, investigate further or scope a separate Proof of Value. The standard commercial model is a fixed fee, with scope, fee, included work and exclusions agreed in writing before the Sprint begins. For a limited founding-partner cohort, SerialLabs may fund all or part of that professional fee from its own business-development budget. Eligibility, the funded amount and every excluded cost are confirmed in the signed scope; sponsorship creates no publicity, data-reuse, productisation or Proof of Value obligation.

The method has seven evidence states and six transitions:

*Observed friction -> Pattern candidate -> Working system -> Measured proof ->
Reusable core -> Owned product -> Platform or venture option*

At each state, the next commitment must be earned. A working system must operate safely in the real workflow. A measured proof must cross a decision threshold fixed before outcomes are observed. A reusable core must survive contextual change. A product requires independent demand, supportability, safe rights, distribution and coherent economics. A platform or venture requires additional ecosystem or organisational evidence. Valid decisions include moving forward, adapting, retaining an internal capability or stopping.

For an operating leader, this has four practical consequences:

- start with one material process and an accountable owner, not an enterprise-wide AI programme;
- agree the baseline, threshold, observation period and decision rule before judging the intervention;
- separate reusable architecture from contextual data, policy, integration and human authority;
- treat productisation as a capital-allocation decision, not as a reward for technical completion.

The paper contributes an operational-pattern anatomy, **P = <C,T,A,I,D,X,O,G>**; a seven-level evidence hierarchy; a core-and-edge architecture; seven productisation gates; provisional falsification thresholds; and records that make evidence, rights and stop decisions explicit. One worked journey instantiates the notation. One documented negative case shows the method stopping a public product claim when the evidence did not support it.

This document presents the complete framework and research programme. **Current editions:** <https://www.seriallabs.com/pattern-to-product>

Abstract

Established companies contain recurring patterns of work, coordination and decision-making that may reveal opportunities for better operating systems and, in narrower circumstances, reusable products. Those patterns are often visible in repeated tasks, recurring exceptions, fragmented information, delayed decisions, manual handoffs and local workarounds. Yet repetition by itself is not a product signal. A recurring problem becomes strategically interesting only when its underlying structure can be distinguished from local context, when a working intervention creates observable value, when a reusable core can be separated from contextual adaptation, and when demand and distribution justify ownership of a product.

This paper develops Pattern-to-Product as a theory-building framework for moving from operational evidence to scalable assets without confusing a successful client project with a market. It synthesises research on organisational routines, knowledge creation, dynamic capabilities, design science, service productisation, modularity, user innovation, platforms and AI governance. It proposes a unit of analysis for an operational pattern, an evidence ladder, a core-and-edge architecture, and distinct gates for systemisation, productisation, platform development and venture creation.

The central claim is deliberately conditional: organisations can be discovery environments for product opportunities, but only when learning is governed, client-specific knowledge remains protected, evidence travels across contexts, and commercial demand exists independently of the original engagement. The framework therefore treats productisation as a decision, not a default destination. It also rejects the assumption that every product should become a platform or company.

The paper concludes with testable propositions, a prospective research programme, an operating model and record templates. Pattern-to-Product is presented as a method used in SerialLabs practice and as a falsifiable managerial theory whose general claims require further empirical testing. It is not a peer-reviewed finding or a claim that SerialLabs has already demonstrated every transition described here.

Keywords: organisational routines; productisation; design science; modularity; AI-enabled work; dynamic capabilities; venture creation; platform strategy; governance; operational learning.

Reader's note: epistemic status

This document has four kinds of statement. First, it summarises established academic or normative work and cites the relevant source. Second, it records the method SerialLabs uses to structure discovery, proof and productisation decisions. Third, it interprets prior work for the Pattern-to-Product model. Fourth, it advances original propositions that require empirical testing. The distinction matters. A practice claim says what method is used; it does not prove that every transition succeeds. A cited theory does not prove the SerialLabs thesis, and an internal example does not establish a general result.

The paper is therefore best read as a structured research and operating agenda. It is not legal advice, an investment memorandum, an assurance report, a certification, or a representation that any illustrative system has achieved a stated business outcome.

1. The proposition

1.1 Opening vignette: an order exception hidden in fragments

Consider an order that will miss a customer commitment. No single event makes it critical. Stock availability changes, a warehouse cut-off is missed, a shipment becomes incomplete and a carrier event arrives late. Each team sees one fragment. The operating problem is not merely prediction; it is the recurrent relationship between signals, priority, ownership, authorised action and resolution.

Written in the Pattern-to-Product notation, the candidate is:

Element	Order Exception Control instantiation
C - Context	A distributor or manufacturer managing customer orders across inventory, warehouse and carrier systems
T - Trigger	A sequence of stock, fulfilment, cut-off, shipment or carrier signals raises the risk of a missed commitment
A - Actors	Supply-chain operations, warehouse, customer service, account owner and the systems that hold the evidence
I - Information	Order lines, stock, promised dates, warehouse events, carrier status, customer importance and prior interventions
D - Decision	Which order needs intervention now, why, and who is accountable?
X - Action	Assign ownership, expedite, replan, escalate or communicate with the customer within defined authority
O - Outcome	Earlier ownership and resolution, fewer avoidable misses, and a recorded explanation of what changed
G - Guardrails	Commercial thresholds, role permissions, escalation rules, evidence lineage and human approval for consequential actions

This example is illustrative, not a client performance claim. Its purpose is to show what the formalism does: it prevents a dashboard, model or interface from being mistaken for the pattern. The reusable hypothesis concerns the relationship among eight elements. Whether that relationship travels is an empirical question.

1.2 The business already contains the opportunity

The conventional search for an AI or software opportunity often begins outside the work: a technology is selected, use cases are brainstormed, and teams then search for somewhere to apply it. Pattern-to-Product reverses that order. It begins with the operating environment and asks what already repeats.

Repeated work leaves signals. An exception returns every week. A report is assembled from the same disconnected sources. Experienced employees repeatedly reconstruct the same context before making a decision. Customer issues follow recognisable trajectories. Different teams build similar spreadsheets, checks

or workarounds. These are not automatically products. They are candidate patterns: evidence that a stable problem structure may exist beneath locally variable performances.

SerialLabs first articulated the proposition internally as follows: businesses already contain the patterns from which their next operating systems and products can be built; those patterns are hidden in repeated tasks, fragmented data, delayed decisions and manual coordination. The public operating sequence later became: recognise, diagnose, build, prove, scale and productise, with productisation permitted only when the problem repeats and distribution is credible.

The thesis can be stated compactly:

A recurring operational problem can become a product opportunity when a working intervention demonstrates value, a reusable solution core survives contextual variation, demand repeats beyond the first setting, and the rights, governance and distribution conditions support ownership.

The verbs in that sentence are intentionally conservative. A problem *can* become an opportunity; it does not inevitably do so. A reusable asset *may* support a product; it does not by itself constitute one. A product *may* create a platform or venture; neither is a mandatory measure of success.

Origin note. The July 2026 Digital Experience Blueprint and the website snapshots archived on 17 and 19 August 2026 establish the provenance and evolution of this thesis; they are not external evidence for its validity. The public records are available at <https://github.com/vitormagalhaes/SerialLabs/tree/website-before-cws-positioning-2026-08-17> and <https://github.com/vitormagalhaes/SerialLabs/tree/website-cws-positioning-production-2026-08-19>.

1.3 Why now

Three changes make the proposition particularly relevant.

First, digital operations produce more observable traces. Event logs, tickets, messages, documents, transactions and system states can reveal how work actually unfolds. Process mining research formalises the use of event data to discover and compare process behaviour, while also warning that data quality, interpretation and context remain essential (van der Aalst et al., 2012).

Second, AI lowers the cost of some cognitive activities: retrieval, classification, extraction, drafting, comparison and case preparation. Rigorous studies show meaningful gains in specific tasks and settings, including professional writing and customer support (Noy and Zhang, 2023; Brynjolfsson, Li and Raymond, 2025). The effects are not uniform. Dell'Acqua et al. (2026) show that performance can improve inside an AI capability frontier and deteriorate outside it. The implication is not simply that AI makes building cheaper. It makes task boundaries, human judgment and workflow design more consequential.

Third, digital architectures are increasingly modular and recombining. Digital components can be rearranged across devices, networks, services and content layers, creating new possibilities for reuse and innovation (Yoo, Henfridsson and Lyytinen, 2010). That generativity increases option value, but it also makes premature claims of reuse easy. A connector used twice is not yet a product. A collection of components is not yet a platform. Technical recombination must be matched by stable demand, operational adoption, governance and distribution.

1.4 The strategic tension

Pattern-to-Product sits between two failure modes.

The first is permanent custom work. Every engagement begins from zero; useful knowledge remains tacit; delivery speed does not improve; similar solutions diverge; and learning is lost when people leave. The second is premature productisation. A one-off success is abstracted too early; customer-specific assumptions are mistaken for universal requirements; the product accumulates exceptions; distribution is assumed rather than demonstrated; and ownership boundaries become unsafe.

The aim is not maximum standardisation. It is disciplined movement from local truth to transferable value. That movement depends on a series of gates that become stricter as commitment increases.

2. Intellectual foundations

Pattern-to-Product is an original synthesis rather than a claim of an entirely new phenomenon. Its components are grounded in several established research traditions.

2.1 Organisational routines are patterns, but not static scripts

Evolutionary theory treats routines as important carriers of organisational capability and memory (Nelson and Winter, 1982). Later routine research rejects the idea that a routine is simply a fixed procedure. Feldman and Pentland (2003) distinguish between the ostensive aspect of a routine - the abstract idea of how it works - and the performative aspect - the specific actions taken by specific people at specific times. Pentland and Feldman (2005) consequently describe routines as generative systems with internal structures and dynamics.

This distinction is foundational. A pattern is not discovered merely by observing that the same label appears repeatedly. "Invoice reconciliation" may contain many different performances, control regimes and exception types. The analyst must identify which relations remain stable and which vary with context. Pattern-to-Product therefore searches for structural invariants inside varied performances, not for identical sequences.

The same logic explains why workarounds matter. A workaround may reveal a broken process, but it may also embody valuable local judgment. Removing it without understanding its function can degrade performance. The pattern is the relationship between context, trigger, information, decision, action and outcome, including the adaptations people use to keep the routine viable.

2.2 Knowledge must move from individual experience to organisational capability

Nonaka (1994) argues that organisations create knowledge through interactions between tacit and explicit knowledge. Cohen and Levinthal (1990) describe absorptive capacity as the ability to recognise the value of external information, assimilate it and apply it. Kogut and Zander (1992) emphasise the firm's ability to recombine knowledge and replicate technology through organising principles that are not reducible to individual information.

Pattern-to-Product uses these insights in a practical way. Experienced operators often know which weak signals matter, which exceptions are dangerous and which sources can be trusted. A working system can help articulate that knowledge, but codification is not neutral. Some judgment cannot be fully represented in rules or models; some should remain under accountable human authority; and some is confidential to the organisation where it was learned.

The relevant capability is therefore not extraction. It is governed articulation: making enough knowledge explicit to improve the work, while preserving context, uncertainty, permissions and decision rights.

2.3 Building can be a method of inquiry

Design science treats the construction and evaluation of artefacts as a way to extend human and organisational capabilities and to generate knowledge about a problem domain (Hevner et al., 2004). Action

design research goes further by treating building, intervention and evaluation as interwoven activities; the artefact is shaped by its organisational context while it is being developed and used (Sein et al., 2011).

This supports the Pattern-to-Product preference for a bounded working system over an abstract recommendation. A working intervention exposes integration constraints, exceptions, behavioural responses, control requirements and adoption friction that a slide deck cannot reveal. It is an epistemic instrument as well as a deliverable.

That does not make every build an experiment. A credible Proof of Value needs a defined question, baseline, observation period, threshold and decision rule. Without these, a functioning artefact may generate enthusiasm but little reliable learning.

2.4 Productisation is standardisation plus intelligibility and exchangeability

Research on professional service productisation is directly relevant. Jaakkola (2011) identifies practices that specify and standardise the offering, make expertise tangible, and systemise processes and methods. Harkonen, Haapasalo and Hanninen (2015) synthesise productisation as the analysis of a need and the combination of tangible and intangible elements into a standardised, repeatable and comprehensible offering.

Pattern-to-Product adopts this definition but adds stronger evidence and governance gates. Productisation is not merely packaging a service or naming a method. In operational software and AI-enabled systems, it requires an architecture that can carry a stable core across contexts, an operating model that can support it, and commercial demand that does not depend entirely on the original relationship.

2.5 Modularity makes reuse possible, but architecture determines what can travel

Henderson and Clark (1990) show that innovation can reside in the architecture connecting components, not only in the components themselves. Schilling (2000) develops a broader theory of modularity, while Yoo, Henfridsson and Lyytinen (2010) show how digital innovation relies on layered modular architectures.

The Pattern-to-Product implication is that reuse should be designed around interfaces and boundaries. A stable product core needs explicit contracts with contextual edges: identity, permissions, data models, integrations, language, policy, thresholds and human roles. If these edges are entangled with the core, each deployment becomes a disguised custom project. If the boundary is too rigid, the product cannot accommodate legitimate variation.

2.6 Learning must balance exploration and exploitation

March (1991) frames the enduring tension between exploration of new possibilities and exploitation of existing knowledge. Dynamic capabilities research similarly focuses on sensing opportunities, seizing them and reconfiguring organisational assets under change (Teece, Pisano and Shuen, 1997).

Pattern discovery is exploratory. Systemisation begins to exploit what has been learned. Productisation increases the exploitation commitment by standardising architecture, offering and delivery. The model needs

both modes: too much exploration produces disconnected prototypes; too much exploitation freezes a weak abstraction and creates path dependence.

2.7 Demand, complementary assets and distribution determine who captures value

Lead-user research shows that users experiencing needs ahead of the broader market can contribute valuable product concepts (von Hippel, 1986). But a strong need and a useful solution do not ensure commercial success. Teece (1986) demonstrates that value capture depends on appropriability and complementary assets such as distribution, manufacturing, service or market access.

For Pattern-to-Product, this creates a hard rule: repeated need is not enough. A productisation decision also requires a credible route to acquisition, implementation, support and trust. Distribution is part of the product thesis, not a post-build marketing task.

2.8 Platforms and ventures require new evidence

Gawer and Cusumano (2014) distinguish internal product platforms, which support families of derivative products, from industry platforms on which external innovators build complements. Platform research emphasises the coevolution of architecture, governance and ecosystem dynamics (Tiwana, Konsynski and Bush, 2010), and the role of modularity and non-generic complementarities in ecosystems (Jacobides, Cennamo and Gawer, 2018).

This literature argues against casual platform language. A product with several modules is not necessarily a platform. A product becomes an internal platform when a common core reliably supports a family of offerings. It becomes an external platform only when third parties can create value through governed complementarities. A venture, in turn, is an organisational and economic commitment: a distinct market, model, team and ownership logic.

3. Definitions and unit of analysis

3.1 What is an operational pattern?

Pattern-to-Product defines an operational pattern as:

A recurrent, evidence-supported relationship among a context, trigger, actors, information, decision, action, outcome and constraints, stable enough for a reusable problem structure and solution core to be distinguished from local variation.

The unit can be represented as:

P = <C, T, A, I, D, X, O, G>

where:

- **C - Context:** the operating environment, objective and boundary.
- **T - Trigger:** the event or condition that starts or changes the work.
- **A - Actors:** people, teams, systems and accountable owners.
- **I - Information:** evidence, data, documents and signals used.
- **D - Decision:** recommendation, judgment or authorisation required.
- **X - Action:** what changes in the operating environment.
- **O - Outcome:** the observable result and follow-up.
- **G - Guardrails:** permissions, policy, risk, escalation and review conditions.

This representation prevents a common category error. Similar interfaces do not prove the same pattern; the decision or action may be different. Similar outcomes do not prove the same pattern; the evidence and constraints may differ. The pattern is relational.

Alexander et al. (1977) use patterns as context-linked design solutions that can be combined into a language. Pattern-to-Product borrows the ideas of a reusable unit and composability, but changes the epistemic role. An operational pattern is initially a descriptive and falsifiable hypothesis about work, not a prescriptive solution template. The P anatomy must be supported by occurrences, counterexamples and outcomes before a solution pattern is standardised. A coherent design language can guide construction; it does not, by itself, demonstrate recurrence, value or demand.

3.2 A pattern is more than frequency

Frequency is one signal of recurrence, but it is neither necessary nor sufficient on its own. A low-frequency, high-consequence exception may justify a system. A high-frequency activity may be too trivial, too variable or too cheap to support a product.

A credible pattern candidate needs four minimum properties:

1. **Recurrence:** the problem or decision arises more than once, or is predictably episodic.
2. **Structural similarity:** the causal or operational relationships are comparable across occurrences.
3. **Materiality:** resolving the pattern can change an outcome that matters.
4. **Observability:** enough evidence exists to distinguish current performance from improvement.

Transferability, codifiability, rights and demand are assessed later. They should not be assumed during discovery.

3.3 Distinctions that protect the thesis

Concept	Working definition	What it is not
Event	A recorded occurrence or state change	A complete process
Task	A bounded unit of work	A product opportunity by itself
Routine	A recognisable pattern of interdependent action	A perfectly fixed sequence
Workflow	An explicit orchestration of tasks, decisions and handoffs	Proof that the workflow is valuable
Use case	A situated description of an actor achieving an outcome	Evidence of repeatable demand
Operational pattern	A recurrent relational structure with observable consequences	A visual similarity or generic industry label
Reusable component	A technical or procedural asset that can be used again	A standalone product
Working system	An intervention operating within a defined context	A validated market
Product	An owned, supportable and repeatable offering for a defined market	A renamed custom project
Platform	A governed common foundation enabling derivative products or complements	A product with many features
Venture	A distinct organisational and economic vehicle	The automatic destination of a product

3.4 Scope conditions

The framework is most plausible where work is recurring, knowledge-intensive, system-mediated, observable and consequential enough to justify measurement. Typical settings include operational exceptions, reconciliation, management intelligence, customer resolution, technical knowledge, compliance workflows and cross-system coordination.

It should not be used as a product-discovery shortcut for every form of professional work. The following are explicit exclusions or stop conditions:

Excluded or weak-fit domain	Why Pattern-to-Product should stop or remain a local method
One-off strategic transactions, litigation events, crisis negotiations or bespoke restructurings	The value depends on singular circumstances, adversarial judgment or relationships that should not be abstracted into a recurrent product claim
Original creative commissions where novelty is the principal value	Repetition and standardisation may destroy rather than preserve the intended outcome
Clinical, employment, credit or other consequential decisions without lawful representative evidence and accountable domain authority	A pattern cannot be tested safely or governed within an acceptable decision boundary
Cheap tasks whose current burden is lower than integration, assurance and change cost	Recurrence is real but economically immaterial
Environments where every context changes the decision anatomy or core controls	The supposed core is a generic label; the useful capability remains bespoke

A market can also be too small to support product ownership even when the operating pattern is real. In that case, the valid outcome is a working system, a reusable internal component or a documented stop decision.

4. The Pattern-to-Product model

The Pattern-to-Product evidence ladder



Commitment increases only when evidence becomes stronger.

At every state, a valid decision is to adapt, remain a capability, or stop.

Figure 1. Seven evidence states connected by six transitions. Each commitment requires stronger evidence; none is automatic.

The model contains seven evidence states connected by six transitions:

Observed friction -> Pattern candidate -> Working system -> Measured proof -> Reusable core -> Owned product -> Platform or venture option

Each transition asks a different question.

1. **Observed friction:** Is there a material operating condition worth understanding?
2. **Pattern candidate:** Does a recurrent structural relationship explain the condition?
3. **Working system:** Can a bounded intervention operate safely in the real workflow?
4. **Measured proof:** Did the intervention change an agreed outcome relative to a credible baseline?
5. **Reusable core:** Which elements remain invariant when context changes?
6. **Owned product:** Do repeated demand, supportability, rights and distribution justify standardisation and ownership?
7. **Platform or venture option:** Do complements, economics, defensibility and organisational separation justify a larger commitment?

The sequence is not a waterfall. Learning can send the work backwards. A build may disprove the assumed pattern. A second deployment may reveal that the core is smaller than expected. Demand may exist but be uneconomic to distribute. A product may remain valuable without becoming a platform or venture.

4.1 The evidence ladder

Stage	Primary question	Minimum evidence	Valid negative decision
Observed friction	Is the condition material?	Examples, owner, frequency or consequence	Ignore or monitor

Stage	Primary question	Minimum evidence	Valid negative decision
Pattern candidate	Does structure repeat?	Comparable occurrences and relational map	Treat as local exception
Working system	Can an intervention operate?	Bounded live or representative run	Redesign or stop
Measured proof	Did the outcome change?	Baseline, threshold, observation, evidence	Adapt or stop
Reusable core	What travels across contexts?	Invariant/variant analysis and second context	Retain as bespoke capability
Owned product	Is there an owned repeatable offering?	Independent demand, support model, rights, economics, distribution	Continue as service/component
Platform or venture option	Does a new coordination or company logic exist?	Complements or distinct market, governance, team and capital logic	Keep as focused product

The negative decisions are essential. A gate that only approves is not a gate; it is a ceremony.

5. Discovering a pattern

5.1 Start from operational symptoms, not product ideas

Pattern discovery begins with a process owner and a bounded operating condition. Useful entry signals include:

- repeated manual comparison, re-keying or reconciliation;
- recurring exceptions with slow ownership or escalation;
- information assembled from multiple systems before the same decision;
- repeated expert questions whose answers depend on dispersed evidence;
- recurring customer requests that require similar diagnosis;
- local spreadsheets or messages acting as unofficial coordination layers;
- delayed reports whose value decays before action;
- repeated demand for the same outcome across teams or organisations.

These signals are prompts for inquiry, not proof of a solution. The team should resist technology-first language until it can describe the current decision and action boundary.

5.2 Use multiple evidence channels

No single channel is reliable enough. Interviews reveal meaning and tacit judgment but may rationalise practice. Event logs reveal sequences but omit unrecorded work. Documents and messages reveal evidence flows but may not show authority. Observation reveals performance but can change behaviour.

A robust diagnostic triangulates:

- process-owner objectives and constraints;
- frontline walkthroughs of real cases;
- event, transaction or ticket histories;
- artefacts such as spreadsheets, reports, documents and messages;
- exception samples, including failed and ambiguous cases;
- decision rights, escalation rules and downstream actions;
- baseline measures and data-quality limitations.

Process mining can support discovery where event logs are fit for purpose, but the model must distinguish recorded process from actual work (van der Aalst et al., 2012). Missing events, inconsistent identifiers and undocumented human coordination are themselves findings.

5.3 Create a Pattern Candidate Record

The first formal artefact is not a product brief. It is a Pattern Candidate Record containing:

- the operational boundary and accountable owner;
- representative occurrences and counterexamples;
- the $P = \langle C, T, A, I, D, X, O, G \rangle$ anatomy;
- known variation by team, geography, customer, system or policy;
- current baseline and confidence in it;
- harm, failure and escalation conditions;
- evidence gaps;
- the smallest intervention that could produce useful learning.

The record must include counterexamples. If the team only documents cases that fit the candidate, it will overstate recurrence and under-design the edge.

5.4 Pattern confidence

Pattern confidence is a structured judgment, not a universal score. It should consider:

- number and independence of observed occurrences;
- similarity of trigger-decision-action relationships;
- stability of the outcome sought;
- variation in inputs, roles and constraints;
- quality and completeness of evidence;
- plausible alternative explanations;
- whether the pattern persists outside one individual or team.

The output is a confidence statement with reasons and unknowns. Numeric scoring may be used locally for prioritisation, but it should not create false precision.

6. From pattern candidate to working system

6.1 Define the decision before the technology

A working system is built around a decision and its consequences. Before selecting a model or interface, the team should be able to state:

- what recommendation or decision is being supported;
- who holds authority;
- what evidence is admissible;
- what action follows;
- what can block, reverse or escalate the action;
- what record must remain after the event.

This is particularly important for AI-enabled work. Task-level productivity does not establish workflow-level value, and AI performance is uneven across superficially similar tasks (Dell'Acqua et al., 2026). The system must make capability boundaries visible.

6.2 Build the smallest evidentiary intervention

The smallest useful system is not necessarily the smallest interface or codebase. It is the smallest intervention capable of answering the decision question. It may include manual review, synthetic data, shadow operation or a narrow live workflow.

The intervention should deliberately preserve observability:

- source evidence remains inspectable;
- model or rule outputs are logged with versions;
- human decisions and overrides are recorded;
- actions are linked to owners and timestamps;
- outcomes can be compared with the baseline;
- uncertainty and missing data are not silently converted into confidence.

Design science supports building as inquiry, while action design research reminds us that artefact and organisational context shape each other (Hevner et al., 2004; Sein et al., 2011). The system is therefore expected to change during the proof period; those changes should be documented as learning, not hidden as implementation noise.

6.3 Proof of Value protocol

A credible Proof of Value should specify before evaluation:

1. **Decision question:** what larger commitment will the evidence inform?
2. **Unit of observation:** case, order, invoice, question, meeting or other bounded unit.
3. **Baseline:** current performance and method used to estimate it.
4. **Intervention:** exact system boundary and human/AI roles.
5. **Outcome measures:** operational, quality, risk and adoption measures.
6. **Threshold:** minimum useful movement and the decision rule. These are fixed after the baseline design is accepted but before the intervention or outcome-observation period begins; the locked version is retained in the evidence record.
7. **Observation period:** long enough to capture representative variation.
8. **Guardrails:** safety, permissions, review and stop conditions.
9. **Decision rule:** scale, adapt or stop.

Where causal attribution matters, the design should use the strongest feasible comparison: randomised assignment, phased rollout, matched cases, interrupted time series or a clearly bounded before/after comparison. When none is feasible, the claim must remain correspondingly modest.

6.4 Evidence hierarchy

The evidence hierarchy

Each level supports a stronger claim. A demonstration proves existence - not value, transfer or a market.

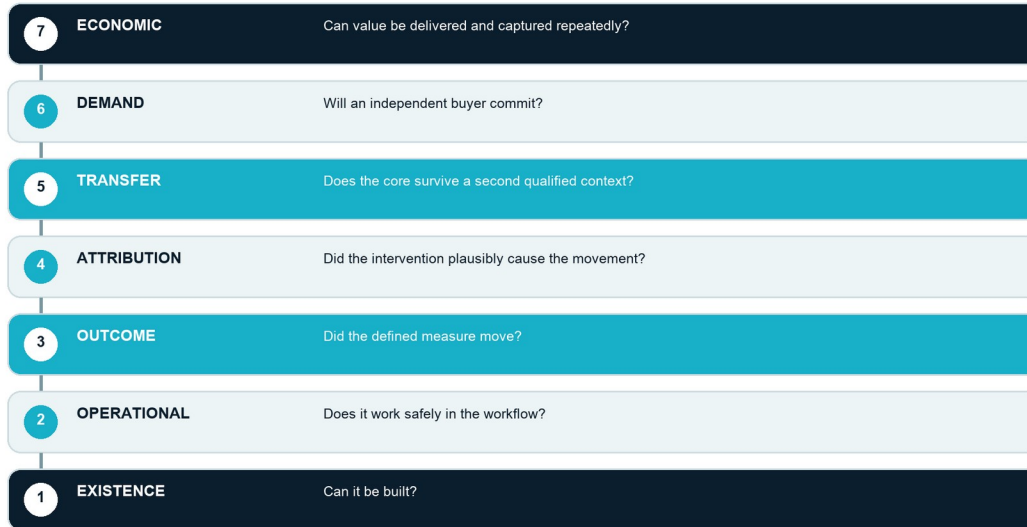


Figure 2. The evidence hierarchy. Each level supports a stronger claim; a demonstration establishes only existence.

The following hierarchy prevents a frequent slide from technical completion to unsupported commercial claim:

- **Existence evidence:** the component or workflow functions.
- **Operational evidence:** it functions in the intended environment.
- **Outcome evidence:** an agreed measure changed.
- **Attribution evidence:** the change can credibly be related to the intervention.
- **Transfer evidence:** the result or core replicates in another context.
- **Demand evidence:** another buyer chooses the outcome under real constraints.
- **Economic evidence:** the offering can be acquired, delivered and supported sustainably.

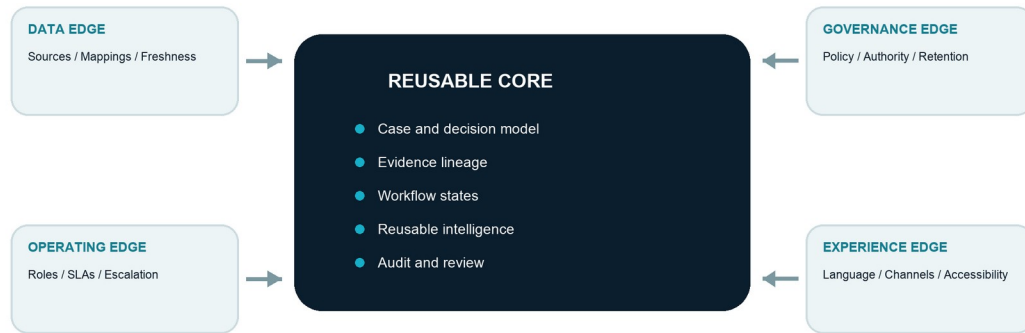
A working demo provides existence evidence. It does not provide the six later forms.

7. From working system to reusable core

7.1 The core-and-edge principle

Core-and-edge abstraction

The core is a transfer hypothesis; the edge contains legitimate contextual variation.



Transfer is demonstrated when the core survives a second context with controlled adaptation and no governance regression.

Figure 3. A reusable core remains stable while data, operating, governance and experience edges adapt to context.

Reusable architecture requires separating what is invariant from what is contextual.

The **core** contains elements expected to remain stable across qualified deployments: canonical workflow states, evidence lineage, case objects, orchestration, audit events, reusable decision-support capabilities, interface primitives and control mechanisms.

The **edge** contains legitimate variation: source-system adapters, identity, data mappings, terminology, local policy, thresholds, approval roles, retention rules, language, user experience and change-management needs.

Layer	Typical core	Typical edge
Problem model	Stable case or decision anatomy	Local definitions and severity
Data	Canonical entities and lineage	Source schemas and mappings
Workflow	State model and orchestration	Roles, SLAs and escalation paths
Intelligence	Reusable extraction, retrieval or ranking capability	Local corpus, prompts, rules and evaluation set
Governance	Audit events, permission model, review states	Policy, legal basis, retention and authority
Experience	Reusable interaction patterns	Vocabulary, channels and accessibility needs
Operations	Observability and release controls	Hosting, integration and support environment

The table is a design hypothesis. The core cannot be declared by architecture alone; it must survive contextual change.

7.2 Reuse is empirical

A component is potentially reusable when it has a clear interface and no known dependency on confidential context. It becomes demonstrably reusable only when used again with less adaptation than rebuilding and without degrading control or outcome quality.

The second context is therefore decisive. It tests whether the abstraction is genuine or whether the first implementation simply concealed assumptions. Useful transfer evidence includes:

- proportion of behaviour delivered by the common core;
- adaptation effort and elapsed time;
- defects and control regressions introduced by variation;
- performance of the same evaluation criteria;
- new edge cases that force core redesign;
- support burden after deployment.

Reuse should improve learning as well as speed. If every deployment produces unstructured exceptions that remain outside the product model, the architecture is accumulating debt rather than evidence.

7.3 Taxonomy of reusable assets

Pattern-to-Product recognises several forms of reuse:

- **Knowledge assets:** pattern records, evaluation sets, taxonomies, decision criteria and failure modes.
- **Technical components:** connectors, parsers, event models, retrieval, orchestration, interfaces and observability.
- **Governance components:** permission patterns, review states, audit records, retention controls and escalation mechanisms.
- **Delivery components:** diagnostic methods, baseline protocols, test harnesses, deployment playbooks and support procedures.
- **Commercial components:** defined outcome, qualification criteria, scope boundary, pricing logic and proof requirements.

The existence of these assets can make later systems faster without requiring a product. Reusable capability is a valuable destination in its own right.

7.4 Client-specific knowledge must not leak into the core

The reusable core must exclude:

- client data and derived confidential information;
- client-specific process logic unless separately licensed;
- unique policy, pricing, customer or operational rules;
- proprietary labels, documents, prompts or evaluation examples;
- implementation knowledge that contractual terms reserve to the client.

Generalisation is not a linguistic exercise. Removing a client name does not make logic generic. The team must establish independent provenance for the pattern and for every reusable element.

8. The productisation gate

Productisation is justified only when all seven dimensions are sufficiently supported. Approval is conjunctive rather than compensatory: exceptional technical reuse cannot compensate for absent demand, and strong demand cannot compensate for unsafe rights.

The model distinguishes **hard gates** from **evidence gates**. Gate 6 - safe ownership and governance - is a hard gate: ambiguous rights, prohibited data use or unacceptable risk blocks productisation and may require the work to stop. The other gates remain conjunctive for a Productise decision, but a candidate may stay in time-bounded incubation while a named, testable evidence gap is resolved. Incubation is not a partial pass and cannot override a hard gate.

8.1 Gate 1 - recurrent problem

Question: Does the same structural problem occur across independent workflows, teams or organisations?

Evidence: comparable Pattern Candidate Records; stable decision and outcome anatomy; counterexamples understood.

Failure condition: recurrence is only superficial, concentrated in one client, or dependent on one individual's practice.

8.2 Gate 2 - demonstrated value

Question: Has a working intervention produced a meaningful outcome under defined conditions?

Evidence: baseline, observation period, threshold, outcome and claim boundary.

Failure condition: only existence or user enthusiasm is demonstrated.

8.3 Gate 3 - stable reusable core

Question: Can a common architecture serve qualified contexts while preserving necessary variation at the edge?

Evidence: second-context transfer, adaptation data, explicit interfaces and versioned evaluation.

Failure condition: each deployment changes the core extensively or creates uncontrolled forks.

8.4 Gate 4 - repeated demand

Question: Do independent buyers or users choose the same outcome strongly enough to support focus?

Evidence: paid pilots, procurement movement, renewal, repeated use, credible willingness to pay or committed design partners.

Failure condition: demand exists only because of a broader consulting relationship, founder persuasion or a single sponsor.

Lead-user evidence is valuable at this stage, but lead users are signals of emerging need, not a substitute for a market (von Hippel, 1986).

8.5 Gate 5 - supportable operating model

Question: Can the offering be implemented, monitored, secured and supported repeatedly?

Evidence: deployment path, service levels, observability, incident response, change management, unit support burden and ownership of the roadmap.

Failure condition: every sale requires exceptional founder intervention or bespoke engineering.

8.6 Gate 6 - safe ownership and governance

Question: Does SerialLabs have the rights and controls required to own and operate the product?

Evidence: provenance register, contractual rights, data boundary, model/provider assessment, risk classification, human authority and exit arrangements.

Failure condition: the reusable proposition relies on client-confidential logic, ambiguous rights, unavailable data or unacceptable risk.

8.7 Gate 7 - credible distribution and economics

Question: Can the product reach, convert and retain customers at an economically coherent cost?

Evidence: defined market, acquisition route, sales cycle, implementation cost, pricing, gross margin hypothesis, retention logic, complementary assets, the fixed product investment I_P , the predeclared horizon N , and the expected non-product and productised delivery-cost ranges T^0_n and T^P_n .

Failure condition: the product can be built but not credibly distributed or supported.

Teece's (1986) complementary-assets argument is crucial here. Technical originality does not determine who captures value. Market access, trust, implementation capacity and support can be decisive.

8.8 Gate decision

The productisation decision

Incubation is a time-bounded evidence state, not a pass around a hard gate.

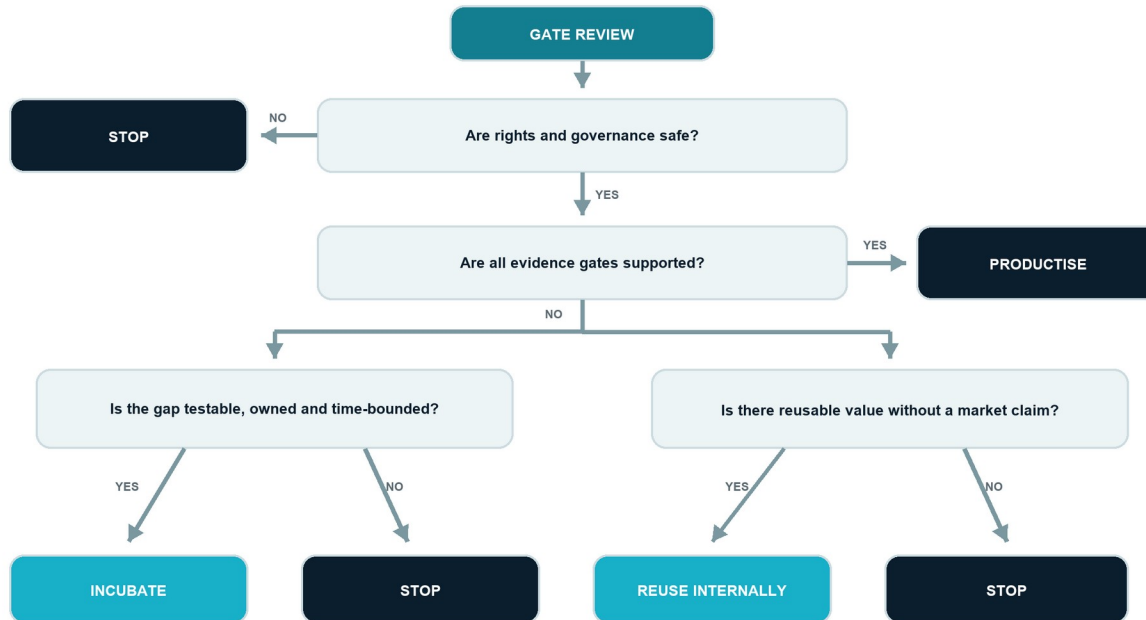


Figure 4. The productisation decision, beginning with a gate review. Incubation is available only for a named, testable and time-bounded evidence gap.

The gate must produce one of four decisions:

- **Productise:** all seven gates are supported, including the hard rights and governance gate.
- **Incubate:** the hard gate is clear, but one or more evidence gates remain unresolved and can be tested by a named experiment.
- **Reuse internally:** retain components and method without a market-facing product.
- **Stop:** do not increase commitment; preserve the learning record.

"Incubate" requires an owner, an expiry date, a specific experiment and a predeclared decision rule. If the evidence is not created by the expiry date, the default is Reuse internally or Stop. Incubation therefore delays a gate decision; it does not compensate for a failed gate.

9. Product, platform and venture are different commitments

9.1 Product

A product is an owned, supportable and repeatable offering for a defined market. Its boundaries, pricing, controls, roadmap and service expectations can be explained without reconstructing a custom statement of work for each buyer.

Product focus often creates more value than platform ambition. A narrow product can deepen workflow fit, evidence and distribution. The model therefore treats "remain a product" as a successful outcome.

9.2 Internal product platform

An internal platform exists when a stable common foundation supports a family of derivative products or modules more efficiently than separate builds. The evidence is architectural and economic: common assets, governed extension points, reduced development effort, coordinated roadmaps and controlled variation (Gawer and Cusumano, 2014).

9.3 External platform

An external platform requires independent complementors that can create value on a governed foundation. Architecture, interfaces, incentives, quality control and ecosystem governance must coevolve (Tiwana, Konsynski and Bush, 2010). Non-generic complementarities and defined roles distinguish an ecosystem from a loose list of partners (Jacobides, Cennamo and Gawer, 2018).

The external platform gate asks:

- Are third parties trying to extend the product for recurring needs?
- Can they do so through stable interfaces without privileged internal access?
- Does the combined value exceed what SerialLabs could efficiently produce alone?
- Are participation, quality, data, liability and value-capture rules governable?
- Do cross-side or same-side effects create durable value rather than complexity?

Without affirmative evidence, "platform" should remain an architectural option, not a market claim.

9.4 Venture

A product becomes a venture candidate when it has:

- a distinct customer and market identity;
- an economic model with a credible path to sustainability;
- a focused roadmap and competitive position;
- defensibility or accumulating advantage;

- a team and governance requirement that cannot remain a side activity;
- an ownership and capital logic appropriate to that separation.

Venture creation is not productisation at larger scale. It creates a new allocation of attention, capital, accountability and risk. Sarasvathy's (2001) effectuation lens is useful under uncertainty: commitments should remain proportionate to available means and affordable loss, while evidence and stakeholder commitments shape the opportunity.

10. Governance, rights and responsible learning

10.1 Governance is part of the architecture

In AI-enabled systems, governance cannot be added after product-market fit. Identity, permissions, evidence lineage, human authority, escalation, monitoring and retention influence the data model, workflow and user experience. They are product requirements.

The NIST AI Risk Management Framework organises AI risk activity around govern, map, measure and manage (Tabassi, 2023). Its Generative AI Profile identifies risks that are novel to or intensified by generative AI and proposes actions across the same lifecycle functions (NIST, 2024). ISO/IEC 42001:2023 establishes requirements for an AI management system. In the European Union, Regulation (EU) 2024/1689 creates risk-based obligations whose applicability depends on the role, system and use context. Pattern-to-Product does not replace these frameworks. It uses a project-specific governance record to ensure that increasing reuse does not silently broaden purpose, users or risk.

As of August 2026, NIST has indicated that AI RMF 1.0 is being revised; organisations should therefore verify the current version and relevant sector-specific requirements before relying on any framework (NIST, 2026).

10.2 The rights boundary

A responsible engagement should define before relevant work begins:

- ownership of client data, processes and pre-existing materials;
- ownership of client-specific deliverables and bespoke development;
- SerialLabs background IP and generic reusable components;
- rights required for the client to use, operate and maintain the deliverable;
- whether any jointly developed or client-specific element may be productised;
- confidentiality, retention, deletion and exit arrangements;
- restrictions on model training, evaluation data and derived artefacts.

The default principle is simple: no quiet productisation. Client confidential information and client-specific logic do not become product assets through exposure or abstraction. Any different commercial model requires explicit written agreement.

10.3 The governance gradient

Governance expands with consequence

Each step adds contexts, dependencies and failure pathways; the geometry is ordinal, not a measured magnitude.

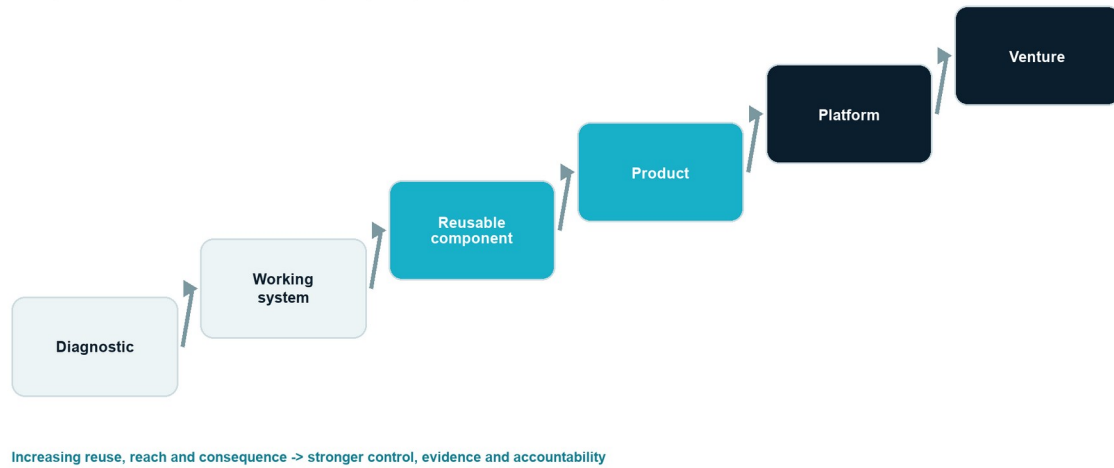


Figure 5. Governance expands ordinally as a diagnostic becomes a system, component, product, platform or venture.

Governance should strengthen as reuse and consequence increase.

- **Diagnostic:** purpose, access, confidentiality and evidence handling.
- **Working system:** roles, permissions, logging, human review and stop conditions.
- **Reusable component:** provenance, testing, versioning and cross-context risk.
- **Product:** lifecycle management, support, security, provider controls and product claims.
- **Platform:** third-party access, ecosystem rules, interoperability and shared risk.
- **Venture:** board accountability, capital allocation, regulatory ownership and independent operating controls.

This gradient counters a common error: treating scale as more of the same. Scale changes the number of affected people, contexts, dependencies and failure pathways.

10.4 Claim governance

Every public performance claim should be traceable to:

- the exact system and version;
- the defined population and period;
- the baseline and comparison;
- the metric definition;

- the observation and attribution method;
- limitations and exclusions;
- approval by the accountable owner.

Representative examples, prototypes and value hypotheses must be labelled as such. A useful internal taxonomy is: **illustrative**, **observed**, **measured**, **attributed**, **replicated** and **independently validated**.

11. The economics of Pattern-to-Product

11.1 Reuse creates option value before it creates product revenue

Reusable assets can reduce discovery time, improve solution quality and make future opportunities cheaper to test. Kogut and Zander (1992) describe how firms build options by recombining accumulated knowledge. Dynamic capabilities similarly emphasise reconfiguration under change (Teece, Pisano and Shuen, 1997).

This option value should not be booked as product success. It is a capability benefit: a larger and better-informed set of feasible future actions.

11.2 A simple cost model

For deployment n , let:

- D_n = discovery and diagnosis effort;
- C_n = common-core engineering effort;
- E_n = edge adaptation and integration effort;
- G_n = governance and assurance effort;
- S_n = support and change effort.
- h_n = material contextual heterogeneity relative to the reference deployment.

Total delivery effort is:

$$T_n = D_n + C_n + E_n + G_n + S_n$$

The identity alone predicts nothing. The productisation hypothesis is directional:

- $D_n + C_n$ should decline as governed knowledge and the common core accumulate;
- that decline should be larger than the incremental rise in $G_n + S_n$ required by stronger product controls and support;
- E_n may respond to heterogeneity h_n , but it should remain bounded by the core-and-edge architecture rather than rise one-for-one with every contextual difference;

- outcome and control performance must remain within the predeclared tolerance. A cheaper transfer that weakens the result or its governance is not learning.

Let T^0_n be the expected total cost of delivering a comparable deployment n through the best credible non-product alternative, estimated before the productisation decision. Let T^P_n be the expected total delivery cost under the productised approach, including discovery, common-core, edge, governance and support effort. Let I_P be the fixed investment required to make the capability an owned product, and let N be the predeclared number of qualified deployments - or equivalent time-bounded deployment horizon - over which the decision will be evaluated.

A necessary delivery-side condition is:

$$\text{sum from } n=1 \text{ to } N \text{ of } (T^0_n - T^P_n) > I_P$$

The comparison must use ranges where uncertainty is material and must not substitute the unusually expensive first deployment for the non-product counterfactual. Gate 7 records T^0_n , T^P_n , I_P , N , the principal assumptions and the sensitivity range before product investment is approved.

This is not a complete valuation model: it excludes revenue, acquisition cost, retention and the timing of cash flows. It states the operating claim that expected delivery savings over a defined horizon must outrun the fixed cost of product ownership while edge effort, outcomes and controls remain acceptable.

Three empirical measures are useful:

- **Core reuse ratio:** proportion of qualified functionality delivered by the unchanged or compatibly extended core.
- **Adaptation ratio:** edge and integration effort divided by total delivery effort.
- **Transfer efficiency:** change in time, cost and defect/control performance relative to a comparable first implementation.

The first two ratios are independent, not complements: core reuse measures a share of qualified **functionality**, whereas adaptation measures a share of delivery **effort**. A deployment can reuse most of the functional core and still consume excessive effort at the edge; that combination falsifies the economic transfer claim even when the architecture appears reusable.

Until a larger dataset supports category-specific calibration, SerialLabs uses the following **provisional falsification convention**, not an industry benchmark:

Measure by the third qualified deployment	Provisional expectation	Falsification response
Core reuse ratio	At least 60% of qualified functionality is delivered by the unchanged or compatibly extended core	Below 60% triggers rejection of the stable-core claim unless a predeclared exception explains the variance
Adaptation ratio	No more than 40% of total delivery effort, and not increasing from the second deployment	Above 40% or a rising trajectory returns the candidate to core-and-edge redesign

Measure by the third qualified deployment	Provisional expectation	Falsification response
Outcome and controls	The agreed outcome stays within tolerance and no material governance regression is introduced	Any material regression blocks a productisation claim regardless of reuse
Transfer efficiency	Positive relative to a comparable first implementation after normalising for scope	No improvement by the third deployment supports Reuse internally or Stop rather than Productise

These numbers make the framework operable and falsifiable. They should be replaced when the governed portfolio provides better empirical calibration; changing them after results are observed is not permitted for the decision already under review.

11.3 Learning curves are conditional

Learning-curve research shows that experience can reduce unit effort, but knowledge can depreciate, transfer imperfectly and vary by organisational setting (Argote and Eppe, 1990). Pattern-to-Product therefore measures whether learning is captured in artefacts, architecture and operating routines rather than assuming repetition automatically creates efficiency.

11.4 Product economics

A product gate requires at least a hypothesis for:

- addressable qualified customers, not generic market size;
- acquisition channel and sales-cycle burden;
- implementation and integration cost;
- recurring infrastructure, model and support cost;
- pricing basis aligned with customer value and usage;
- gross-margin path after realistic support;
- retention, expansion and replacement risk;
- cost of governance, assurance and regulatory change.

The economics should be stress-tested against a service-heavy scenario. If margin depends on classifying recurring custom work as product revenue, productisation has not occurred.

11.5 Portfolio economics

A Pattern-to-Product portfolio should not maximise the number of products. It should maximise the quality of evidence-weighted options while limiting correlated risk and organisational distraction.

Useful portfolio categories are:

- **Capability assets:** reusable internally, no independent product thesis.
- **Incubations:** named product hypotheses with unresolved gates.
- **Focused products:** supported offerings with clear markets.
- **Platform options:** products with emerging derivative or complementor evidence.
- **Venture candidates:** products whose market and organisation justify separation.

Each category needs a different funding horizon, evidence expectation and stop rule.

12. Pattern-to-Product and Cognitive Workflow Systems

Pattern-to-Product and Cognitive Workflow Systems operate at different levels.

Pattern-to-Product is a company-building and asset-creation logic. It explains how SerialLabs moves from recurring operational evidence to reusable capability, product and possible venture.

A Cognitive Workflow System is a specific systems thesis. It addresses a recurring organisational gap between AI-supported cognition and accountable action: a system may retrieve, analyse or recommend, but the organisation still needs an explicit decision boundary, authorised intent, operational execution and review.

Pattern-to-Product governs how SerialLabs uses the CWS design thesis in practice. SerialLabs uses Cognitive Workflow Systems as a system-design discipline in current work: CWS connects cognition to accountable action inside a governed decision boundary. Its repeatable transfer across materially different contexts as a product category remains an empirical hypothesis.

- **Observed friction:** recommendations do not reliably become governed action.
- **Working-system practice:** bounded workflows connect evidence, recommendation, authority, action and review.
- **Reusable-core hypothesis:** decision-boundary records, governed intent, orchestration, evidence lineage and action logs may transfer.
- **Productisation question:** do independent buyers value the same governed foundation enough to support an owned product?
- **Platform option:** only repeated derivative or complementor demand would justify a common external foundation.

The client-facing claim is therefore narrower and stronger: SerialLabs uses the method to design and judge bounded working systems. Product, category and platform claims require their own evidence; a client is not asked to become the test of an unproven platform thesis.

13. Testable propositions

The thesis becomes useful as research only when it can be contradicted. The following propositions define an initial empirical agenda.

Proposition 1 - structural recurrence

Claim: Material operational problems that appear different at the surface can exhibit stable trigger-decision-action-outcome structures across contexts.

Supporting evidence: independently coded cases converge on the same P anatomy with bounded variation.

Falsifier: similarities disappear once context and causal relations are examined, leaving only generic labels.

Proposition 2 - working systems improve pattern knowledge

Claim: Building and operating a bounded intervention reveals material constraints and variation that diagnosis alone does not capture.

Supporting evidence: the pattern record changes in specific, decision-relevant ways after operation; newly discovered constraints affect architecture or claims.

Falsifier: operational use produces no additional reliable knowledge or the learning is attributable only to more elapsed analysis time.

Proposition 3 - a stable core can survive contextual change

Claim: For qualified patterns, a reusable core can be separated from contextual edges and transferred with less effort than rebuilding.

Supporting evidence: second and later deployments preserve core interfaces, reduce comparable effort and maintain outcome/control performance.

Falsifier: adaptation dominates total effort, core changes are extensive, or control quality declines.

Proposition 4 - reuse compounds selectively

Claim: Accumulated components, methods and evaluation assets reduce time-to-evidence for related opportunities.

Supporting evidence: comparable diagnostics and proofs become faster or more reliable after controlling for scope and team experience.

Falsifier: apparent acceleration is explained by narrower scope, lower quality or unrecorded expert effort.

Proposition 5 - product demand must become independent

Claim: A genuine product opportunity generates repeated demand that is not dependent on the original client or broader consulting relationship.

Supporting evidence: independent buyers commit under a common offering and pricing logic.

Falsifier: interest disappears without bespoke scope, founder access or cross-subsidy.

Proposition 6 - governance can be reusable

Claim: Some governance capabilities - evidence lineage, permission patterns, review states and audit events - can form part of a reusable core while local authority and policy remain at the edge.

Supporting evidence: controls transfer across contexts with explicit configuration and no material rights or risk regression.

Falsifier: governance requirements are so context-specific that the common control model creates unsafe simplification.

Proposition 7 - distribution is a co-determinant of productisation

Claim: Products with credible complementary assets and distribution paths are more likely to create sustainable value than technically reusable solutions without them.

Supporting evidence: acquisition, implementation and retention performance explains product viability beyond technical reuse.

Falsifier: technical reuse alone consistently predicts viable product economics.

Proposition 8 - stronger commitments require stronger evidence

Claim: Explicit evidence gates reduce premature scaling and improve the quality of capital and roadmap decisions.

Supporting evidence: gate records predict later adapt/stop decisions, reduce sunk-cost escalation and improve claim accuracy.

Falsifier: gates become bureaucratic rituals, delay useful action without improving decisions, or are systematically overridden.

14. Research programme

14.1 Research design

A credible validation programme should combine retrospective and prospective work.

Phase 1 - retrospective pattern coding. Select a diverse set of completed or active systems, including successes, failures and ambiguous cases. Independently code the P anatomy, evidence level, reusable assets and gate outcomes. Assess inter-rater agreement and revise definitions.

Phase 2 - prospective pattern registry. Register new candidates before building. Record hypotheses, counterexamples, baseline quality, smallest intervention and stop conditions. This reduces hindsight bias.

Phase 3 - transfer studies. For selected candidates, test the same core in a second and third context. Measure adaptation, time, defects, control performance and outcome comparability.

Phase 4 - productisation experiments. Test offering comprehension, independent demand, pricing, acquisition and support. Separate design-partner learning from evidence of a repeatable market.

Phase 5 - longitudinal portfolio study. Track decisions, costs, product outcomes and stopped opportunities over 18 to 36 months. Evaluate whether the gate model improves capital allocation and learning.

14.2 Sampling

Sampling should avoid only studying promising opportunities. The registry should include:

- patterns that remain bespoke;
- proofs that fail to reach the threshold;
- reusable components without product demand;
- products with demand but weak economics;
- products that remain focused;
- any genuine platform or venture transitions.

Negative cases are necessary to estimate base rates and understand boundary conditions.

14.3 Methods

Appropriate methods may include comparative case study, process tracing, event-log analysis, action design research, pre/post or phased comparisons, customer discovery with commitment evidence, architecture conformance analysis and longitudinal economic measurement.

Where quantitative inference is weak, the paper should prefer transparent qualitative evidence to pseudo-precise scoring. Where sufficient observations exist, hypotheses and thresholds should be preregistered internally before results are reviewed.

14.4 Research dataset

The minimum governed dataset includes:

- de-identified pattern ID and context class;
- recurrence and materiality evidence;
- P anatomy and variation;
- baseline definition and quality;
- intervention and system version;
- outcome and control measures;
- human decisions, overrides and stop events;
- reusable assets and provenance;
- transfer effort and performance;
- demand commitments and commercial terms category;
- gate decisions and later outcomes.

Access must follow contractual, privacy and security boundaries. Cross-client research should use de-identified structural metadata only where rights allow it.

15. Operating model for SerialLabs

15.1 Roles

- **Process owner:** accountable for the operating outcome in the client or product context.
- **Pattern lead:** maintains the candidate definition, counterexamples and transfer hypothesis.
- **Evidence steward:** owns baseline, measures, claim level and observation record.
- **Architecture lead:** owns core/edge boundaries, interfaces and conformance.
- **Governance owner:** owns rights, data boundary, model/provider risk, human authority and controls.
- **Product owner:** owns market, offering, distribution, support and product economics after incubation.
- **Pattern Council:** makes cross-functional gate decisions and allocates evidence-building capacity.

One person may hold several roles in an early project, but the responsibilities should remain explicit.

15.2 Required artefacts

Every qualified opportunity should accumulate a consistent evidence trail:

1. Pattern Candidate Record.
2. Proof of Value protocol.
3. Evidence and claim log.
4. Core-and-edge map.
5. Reuse and provenance register.
6. Productisation gate memo.
7. Platform or venture memo, only when relevant.
8. Stop or archive record.

The artefacts should be concise enough to maintain. Their purpose is decision quality and organisational memory, not documentation volume.

15.3 Pattern Council cadence

A monthly Pattern Council can review a small portfolio using four questions:

- What new evidence was created?
- Which assumption became weaker or stronger?
- What is the next minimum commitment?

- What should stop, remain reusable, become a product or graduate to a new gate?

Quarterly, the Council should review calibration: Were past confidence statements accurate? Did incubations expire? Did reuse actually reduce effort? Were any claims broader than the evidence?

15.4 Pattern registry

The registry is not an idea backlog. It is a structured map of evidence-bearing opportunities. Each entry should show status, owner, confidence statement, rights boundary, next experiment, expiry date and links to supporting artefacts.

Sensitive client information should remain in segregated engagement records. The cross-company registry should contain only authorised, abstracted metadata.

16. Illustrative pattern journeys

The following journeys are worked applications based on representative SerialLabs system designs. The Order Exception Control vignette in section 1.1 instantiates the complete P anatomy. Sections 16.1-16.3 are not published client performance results. Section 16.4 is different: it records a verifiable internal stop decision using the public version history of the SerialLabs thesis.

16.1 Order Exception Control

Candidate pattern. An order rarely becomes critical in one moment. Risk emerges through a sequence of stock mismatches, missed cut-offs, incomplete shipments, carrier delays and customer commitments. Different teams see fragments; ownership arrives late.

Working system. Connect order, inventory, warehouse and carrier events; create a prioritised case; show contributing signals, commercial context and probable cause; assign accountable action; observe resolution.

Potential reusable core. Event normalisation, exception case model, risk evidence, prioritisation, ownership state, decision record and outcome learning.

Contextual edge. Order categories, inventory policy, carrier events, service commitments, escalation roles and commercial thresholds.

Productisation evidence required. Similar exception anatomy across independent organisations; earlier detection and ownership relative to baseline; stable core under different systems; repeated willingness to pay; supportable integrations and distribution.

Reason to stop. If each organisation's exception logic is inseparable from its ERP customisation and local operating model, the opportunity may remain a system capability rather than a product.

16.2 Industrial Knowledge Navigator

Candidate pattern. Technicians repeatedly search fragmented manuals and intervention histories or call the expert who remembers. The need is not generic question answering; it is supported guidance with source, permission, currency and safe escalation.

Working system. Retrieve from approved sources, cite evidence, respect permissions, distinguish validated guidance from historical context, identify contradictions, and route uncertain questions to an expert.

Potential reusable core. Permission-aware retrieval, evidence citation, authority states, contradiction detection, escalation workflow and reviewable knowledge capture.

Contextual edge. Equipment family, source corpus, safety policy, terminology, identity system, expert ownership and validation rules.

Productisation evidence required. Transfer across controlled knowledge domains; answer and citation quality on approved evaluation sets; safe refusal; reduced avoidable search or escalation; repeat demand; defensible integration and governance.

Reason to stop. If source quality and expert governance cannot be established, a polished interface would amplify uncertainty rather than create a trustworthy product.

16.3 Cognitive Workflow Systems

Candidate pattern. AI can produce analysis or recommendations, but action remains informally coordinated. Decision authority, constraints and follow-up are missing from the system boundary.

Working system. Represent inquiry, evidence, recommendation, decision boundary, authorised intent, operational action and review as one governed workflow.

Potential reusable core. Evidence objects, decision records, intent schema, action orchestration, permissions, audit and review.

Contextual edge. The substantive decision, organisational authority, sector regulation, action systems, policy and risk thresholds.

Productisation evidence required. Recurrence across materially different workflows; measurable reduction in decision/action failure or coordination cost; stable governance primitives; independent demand for the common foundation.

Reason to stop. If the common model remains too abstract to reduce deployment effort or improve outcomes, CWS may be better used as a design lens than a standalone product category.

16.4 A documented stop: Pattern-to-Product as a public product claim

Candidate. An early SerialLabs thesis connected recurring operational problems to systems, products and possible ventures. The public operating sequence - recognise, diagnose, build, prove, scale and productise - could have been promoted immediately as a named product category or a proven company-building engine.

Evidence available. The July 2026 blueprint documented the thesis, and the archived 17 August website encoded the sequence and productisation conditions. Those artefacts established provenance and strategic intent. They did not establish cross-context transfer, independent demand, supportable economics or a completed productisation transition.

Gate review. The recurrence hypothesis was plausible, but Gates 2-5 and 7 lacked publishable evidence. The issue was not that the thesis had been disproved. It was that an owned-product claim would have exceeded the evidence. Citing the blueprint or the website itself could not close those gaps.

Decision. SerialLabs stopped short of presenting Pattern-to-Product as a validated product or market category. The 19 August positioning release separated Cognitive Workflow Systems from the broader company-building thesis, while Pattern-to-Product was retained as an operating method and research programme. In gate language, the decision was **Reuse internally**, not **Productise**.

What this negative case establishes. This is a low-risk stop about claim scope, not a built and measured system that failed after material delivery investment. It therefore cannot demonstrate that SerialLabs will stop when sunk cost, client expectation or technical attachment is substantial. It demonstrates one narrower

observable practice: claim scope was reduced when provenance was stronger than proof. A stronger negative case would document a built and measured system that failed a gate; none is publishable in this version. The case also explains why internal strategy documents and archived website versions appear as origin notes rather than external authorities in this paper.

Reopening condition. A stronger public product claim would require at least replicated working-system evidence, a stable core across qualified contexts, independent demand and a credible operating and distribution model. Until then, the method may guide work without being sold as an accomplished transition.

17. Failure modes and counterarguments

17.1 Survivorship bias

If the organisation studies only patterns that became products, the theory will appear stronger than it is. The registry must include stopped candidates and failed transfers.

17.2 Abstraction error

Teams may strip away the very context that makes a process work. Core-and-edge analysis must preserve causal and governance dependencies, not merely common screens or fields.

17.3 Premature standardisation

Early standardisation can freeze a weak model and shift cost to exceptions. Incubation should prioritise learning across contexts before optimising delivery.

17.4 Distribution fallacy

Because a problem repeats does not mean buyers recognise it, budget for it or can purchase a common solution. Distribution must be tested as an independent system.

17.5 Integration gravity

B2B products often become service-heavy because identity, data and workflow integration dominate implementation. If adapters cannot contain that variation, product economics may not emerge.

17.6 Governance drag

Controls can become bureaucratic and slow. The answer is proportional governance with explicit decisions and reusable controls, not removal of accountability. Gate effectiveness should itself be measured.

17.7 IP contamination

Cross-client learning can create ethical and contractual risk. Provenance, segregation and explicit rights are necessary; de-identification alone is insufficient.

17.8 AI capability volatility

Model capabilities, costs and provider conditions change. A product core tied too closely to one model may age rapidly. Evaluation, portability and human authority should be architectural concerns.

17.9 Platform inflation

Calling a modular product a platform can encourage unnecessary APIs, ecosystem programmes and governance before complementor demand exists. Platform status must be evidenced.

17.10 Organisational distraction

Product incubation can compete with client delivery and dilute focus. Portfolio limits, expiry dates and separate ownership become necessary as options multiply.

17.11 The market-first objection

The strongest objection is not a failure in execution. It is that Pattern-to-Product may be dominated by a simpler route: select a market, validate independent demand and build a product for it. A market-first path avoids confusing one client's problem with a category, forces an explicit buyer and distribution thesis early, and reduces the risk that service revenue disguises weak product economics. In many software categories it is the better starting point.

Pattern-to-Product does not claim a superior base rate for services-to-product transitions. It makes a narrower comparative claim: operational work can be a high-information discovery environment when the problem is latent inside workflows, value depends on integration and authority, or users cannot evaluate the intervention credibly before it operates. Lead users can expose needs earlier than the broader market (von Hippel, 1986), but complementary assets and independent demand still determine whether that learning becomes a viable product (Teece, 1986).

The routes should therefore be selected, not blended rhetorically:

Prefer market-first when	Prefer Pattern-to-Product when
The buyer, budget, category and desired outcome are already legible	The material problem is visible only through recurring work, exceptions or coordination failure
Demand can be tested cheaply without deep workflow integration	A bounded working system is required to discover the real decision, data and governance boundary
A coherent minimum product can be built without client-specific rights or infrastructure	Operational access creates learning that interviews and market surveys cannot reliably provide
Distribution learning is the main uncertainty	Problem structure, transferability and outcome evidence are the main uncertainties

If independent demand can be tested before building at materially lower cost and risk, market-first dominates. If operational learning is necessary, Pattern-to-Product may justify the first system - but it still must pass the independent-demand gate. Effectuation under uncertainty can explain staged commitments; it does not excuse the absence of a market (Sarasvathy, 2001).

18. Managerial implications

For established companies

The framework offers a way to begin with one recurring process while retaining strategic optionality. The company can obtain a useful system even if no product emerges. It should expect transparent ownership, baseline, human authority and evidence boundaries.

For service and transformation firms

The model distinguishes legitimate capability accumulation from covert reuse of client work. Firms can build generic methods and components, but product claims require independent recurrence, rights, demand and economics.

For product leaders

Operational deployments can be superior discovery environments when they expose real constraints. The challenge is to avoid allowing early customers to define an incoherent core. Product leaders need explicit invariant/variant analysis and independent demand tests.

For investors and venture builders

Pattern-to-Product creates a diligence trail from operating evidence to market commitment. It can reduce narrative inflation by showing which gates are passed, which remain hypotheses and what would falsify the opportunity.

For AI governance leaders

The model treats governance as product architecture and organisational design. Risk, authority and evidence should travel with the workflow as it moves from pilot to reusable product.

19. Conclusion

Pattern-to-Product begins from a modest observation: repeated work contains information. Recurring exceptions, delayed decisions, fragmented evidence and manual coordination can reveal stable problem structures. Building a bounded system can make those structures more visible and create measurable improvement.

The ambitious part of the thesis is not that repetition exists. It is that disciplined operational learning can sometimes create transferable assets, focused products and new companies. That claim is plausible but conditional. The pattern must be real; the result must be measured; the core must survive context; demand must become independent; rights and governance must be safe; distribution and economics must be credible.

The final sequence is therefore not a promise. It is an evidence architecture:

*Observed friction -> Pattern candidate -> Working system -> Measured proof ->
Reusable core -> Owned product -> Platform or venture option*

The word *option* matters. Good strategy preserves the right to continue without creating an obligation to scale. Pattern-to-Product is strongest when it helps SerialLabs build what evidence can support, reuse what legitimately travels, productise only what deserves focus, and stop what should not become larger.

20. Put the method to work

The practical first commitment is not a product build. It is an **AI Opportunity Sprint** around one recurring operational process.

Commitment	Defined boundary
Scope	One material process and one bounded decision, not an enterprise-wide transformation programme
Duration	2-3 weeks
Required participation	A named operating or financial owner, access to the people closest to the work, and agreed representative examples or available data
Deliverables	Current-state diagnosis, Pattern Candidate Record, evidence and rights gaps, initial P anatomy, and a scoped recommendation
Decision	Stop, investigate a named gap, reuse a capability, or scope a separate paid Proof of Value
Standard commercial model	A fixed fee is defined and agreed in writing before work begins. The signed scope states the fee, included work, exclusions, expenses, taxes, payment terms and the boundary between the Sprint and any later Proof of Value
Founding-partner sponsorship	For a limited cohort, SerialLabs may fund all or part of the Sprint professional fee from its own business-development budget. Eligibility requires one material process, an accountable operating or financial owner, timely access to representative evidence and participants, a credible decision the Sprint can inform, portfolio fit and capacity for method learning
Sponsorship limits	The signed scope states the amount or percentage funded. Unless expressly included, sponsorship does not cover client internal time, third-party software or data, travel, procurement charges or extraordinary data preparation. Availability is capacity- and budget-limited and may close when the cohort is full
No implied exchange	Sponsorship creates no obligation to provide publicity, transfer confidential data or IP, permit cross-client reuse, commission a Proof of Value, adopt a SerialLabs product or participate in a later productisation programme

The client-specific diagnosis and records remain governed by the signed engagement terms whether or not the work continues. A Sprint does not promise a product opportunity. It promises a better-evidenced next decision.

Start with the operating context, not a technology brief: <https://www.seriallabs.com/> or info@seriallabs.com.

Appendix A. Pattern Candidate Record

Identification

- Pattern ID and working name
- Date, version and pattern lead
- Process owner and operating boundary
- Confidentiality and rights classification

Pattern anatomy

- Context
- Trigger
- Actors and accountable owner
- Information and source quality
- Recommendation or decision
- Authorised action
- Outcome and follow-up
- Guardrails, escalation and review

Recurrence evidence

- Representative occurrences
- Counterexamples
- Frequency or consequence
- Structural invariants
- Known variation
- Alternative explanations

Evidence state

- Current baseline and confidence
- Materiality
- Data gaps
- Pattern confidence statement

- Claim level: illustrative, observed, measured, attributed, replicated or independently validated

Next commitment

- Smallest evidentiary intervention
- Success and stop conditions
- Owner, timebox and required access

Appendix B. Productisation Gate Record

Gate	Decision question	Evidence	Gap or counter-evidence	Status
Recurrent problem	Does the structure repeat independently?			
Demonstrated value	Did a bounded intervention change a material outcome?			
Reusable core	Has the core transferred with controlled adaptation?			
Repeated demand	Are independent buyers choosing the same outcome?			
Operating model	Can delivery and support repeat?			
Rights and governance	Can the product be owned and operated safely?			
Distribution and economics	Is there a credible route to sustainable value?			

Decision: Productise / Incubate until [date] / Reuse internally / Stop **Decision owner:** **Next evidence event:**
Approved claim boundary:

Appendix C. Measurement catalogue

Pattern measures

- independent occurrences and contexts;
- structural similarity and meaningful variation;
- exception and counterexample rate;
- baseline completeness and confidence.

System measures

- time from trigger to detection;
- time from detection to accountable decision;
- manual touches and handoffs;
- outcome quality and error rate;
- override, escalation and safe-refusal behaviour;
- adoption and actual use in the workflow.

Reuse measures

- core reuse ratio;
- adaptation ratio;
- deployment time and effort;
- defects and control regressions;
- evaluation performance across contexts;
- support burden and fork count.

Demand and economics measures

- independent qualified demand;
- paid commitments and renewal;
- sales-cycle and implementation burden;
- pricing and gross-margin path;
- retention, expansion and support cost;
- dependence on founder or consulting pull.

Governance measures

- evidence lineage completeness;
- permission and access exceptions;
- human review and override traceability;
- incidents, near misses and stop events;
- retention and deletion performance;

- accuracy of public and internal claims.

Glossary

Core: the stable architecture, knowledge and controls expected to transfer across qualified deployments.

Edge: the contextual configuration and adaptation required for a specific organisation or workflow.

Evidence gate: an explicit decision point at which stronger commitment requires specified evidence.

Pattern candidate: a provisional hypothesis that a recurrent operational structure exists.

Productisation: the process of defining, standardising, systemising and making an offering repeatable and supportable for a market.

Proof of Value: a bounded intervention evaluated against an agreed baseline and decision threshold.

Reusable asset: a knowledge, technical, governance, delivery or commercial element that can legitimately support later work.

Working system: an intervention operating within a defined organisational context and decision boundary.

References

- Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, I. and Angel, S. (1977). *A Pattern Language: Towns, Buildings, Construction*. Oxford University Press.
- Argote, L. and Epple, D. (1990). Learning curves in manufacturing. *Science*, 247(4945), 920-924. <https://doi.org/10.1126/science.247.4945.920>
- Brynjolfsson, E., Li, D. and Raymond, L. (2025). Generative AI at Work. *The Quarterly Journal of Economics*, 140(2), 889-942. <https://doi.org/10.1093/qje/qjae044>
- Cohen, W. M. and Levinthal, D. A. (1990). Absorptive capacity: A new perspective on learning and innovation. *Administrative Science Quarterly*, 35(1), 128-152. <https://doi.org/10.2307/2393553>
- Dell'Acqua, F., McFowland III, E., Mollick, E. R., Lifshitz-Assaf, H., Kellogg, K. C., Rajendran, S., Kraye, L., Candelon, F. and Lakhani, K. R. (2026). Navigating the jagged technological frontier: Field experimental evidence of the effects of artificial intelligence on knowledge worker productivity and quality. *Organization Science*, 37(2), 403-423. <https://doi.org/10.1287/orsc.2025.21838>
- European Parliament and Council of the European Union (2024). Regulation (EU) 2024/1689 laying down harmonised rules on artificial intelligence. *Official Journal of the European Union*, 12 July 2024. <https://eur-lex.europa.eu/eli/reg/2024/1689/oj>
- Feldman, M. S. and Pentland, B. T. (2003). Reconceptualizing organizational routines as a source of flexibility and change. *Administrative Science Quarterly*, 48(1), 94-118. <https://doi.org/10.2307/3556620>
- Gawer, A. and Cusumano, M. A. (2014). Industry platforms and ecosystem innovation. *Journal of Product Innovation Management*, 31(3), 417-433. <https://doi.org/10.1111/jpim.12105>
- Harkonen, J., Haapasalo, H. and Hanninen, K. (2015). Productisation: A review and research agenda. *International Journal of Production Economics*, 164, 65-82. <https://doi.org/10.1016/j.iipe.2015.02.024>
- Henderson, R. M. and Clark, K. B. (1990). Architectural innovation: The reconfiguration of existing product technologies and the failure of established firms. *Administrative Science Quarterly*, 35(1), 9-30. <https://doi.org/10.2307/2393549>

- Hevner, A. R., March, S. T., Park, J. and Ram, S. (2004). Design science in information systems research. *MIS Quarterly*, 28(1), 75-105. <https://doi.org/10.2307/25148625>
- ISO/IEC (2023). *ISO/IEC 42001:2023 Information technology - Artificial intelligence - Management system*. International Organization for Standardization. <https://www.iso.org/standard/42001.html>
- Jaakkola, E. (2011). Unraveling the practices of productization in professional service firms. *Scandinavian Journal of Management*, 27(2), 221-230. <https://doi.org/10.1016/j.scaman.2011.03.001>
- Jacobides, M. G., Cennamo, C. and Gawer, A. (2018). Towards a theory of ecosystems. *Strategic Management Journal*, 39(8), 2255-2276. <https://doi.org/10.1002/smj.2904>
- Kogut, B. and Zander, U. (1992). Knowledge of the firm, combinative capabilities, and the replication of technology. *Organization Science*, 3(3), 383-397. <https://doi.org/10.1287/orsc.3.3.383>
- March, J. G. (1991). Exploration and exploitation in organizational learning. *Organization Science*, 2(1), 71-87. <https://doi.org/10.1287/orsc.2.1.71>
- Nelson, R. R. and Winter, S. G. (1982). *An Evolutionary Theory of Economic Change*. Harvard University Press.
- NIST (2024). *Artificial Intelligence Risk Management Framework: Generative Artificial Intelligence Profile*. NIST AI 600-1. National Institute of Standards and Technology. <https://doi.org/10.6028/NIST.AI.600-1>
- NIST (2026). *Artificial Intelligence Risk Management Framework*. Current framework status and resources, accessed 26 August 2026. <https://www.nist.gov/itl/ai-risk-management-framework>
- Nonaka, I. (1994). A dynamic theory of organizational knowledge creation. *Organization Science*, 5(1), 14-37. <https://doi.org/10.1287/orsc.5.1.14>
- Noy, S. and Zhang, W. (2023). Experimental evidence on the productivity effects of generative artificial intelligence. *Science*, 381(6654), 187-192. <https://doi.org/10.1126/science.adh2586>
- Pentland, B. T. and Feldman, M. S. (2005). Organizational routines as a unit of analysis. *Industrial and Corporate Change*, 14(5), 793-815. <https://doi.org/10.1093/icc/dth070>
- Sarasvathy, S. D. (2001). Causation and effectuation: Toward a theoretical shift from economic inevitability to entrepreneurial contingency. *Academy of Management Review*, 26(2), 243-263. <https://doi.org/10.5465/AMR.2001.4378020>
- Schilling, M. A. (2000). Toward a general modular systems theory and its application to interfirm product modularity. *Academy of Management Review*, 25(2), 312-334. <https://doi.org/10.5465/amr.2000.3312918>
- Sein, M. K., Henfridsson, O., Purao, S., Rossi, M. and Lindgren, R. (2011). Action design research. *MIS Quarterly*, 35(1), 37-56. <https://doi.org/10.2307/23043488>
- Tabassi, E. (2023). *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*. NIST AI 100-1. National Institute of Standards and Technology. <https://doi.org/10.6028/NIST.AI.100-1>
- Teece, D. J. (1986). Profiting from technological innovation: Implications for integration, collaboration, licensing and public policy. *Research Policy*, 15(6), 285-305. [https://doi.org/10.1016/0048-7333\(86\)90027-2](https://doi.org/10.1016/0048-7333(86)90027-2)
- Teece, D. J., Pisano, G. and Shuen, A. (1997). Dynamic capabilities and strategic management. *Strategic Management Journal*, 18(7), 509-533. [https://doi.org/10.1002/\(SICI\)1097-0266\(199708\)18:7%3C509::AID-SMJ882%3E3.0.CO;2-Z](https://doi.org/10.1002/(SICI)1097-0266(199708)18:7%3C509::AID-SMJ882%3E3.0.CO;2-Z)
- Tiwana, A., Konsynski, B. and Bush, A. A. (2010). Platform evolution: Coevolution of platform architecture, governance, and environmental dynamics. *Information Systems Research*, 21(4), 675-687. <https://doi.org/10.1287/isre.1100.0323>
- van der Aalst, W. et al. (2012). Process Mining Manifesto. In F. Daniel, K. Barkaoui and S. Dustdar (eds.), *Business Process Management Workshops*, LNBIP 99, 169-194. Springer. https://doi.org/10.1007/978-3-642-28108-2_19
- von Hippel, E. (1986). Lead users: A source of novel product concepts. *Management Science*, 32(7), 791-805. <https://doi.org/10.1287/mnsc.32.7.791>
- Yoo, Y., Henfridsson, O. and Lyytinen, K. (2010). The new organizing logic of digital innovation: An agenda for information systems research. *Information Systems Research*, 21(4), 724-735. <https://doi.org/10.1287/isre.1100.0322>



ABOUT SERIALLABS

SerialLabs is an independent, founder-led company that builds operational systems for established companies, develops products of its own and helps promising maker-built products become stronger. Strategy, product, engineering, data and AI execution operate as one accountable delivery unit.

CONTACT

seriallabs.com | info@seriallabs.com