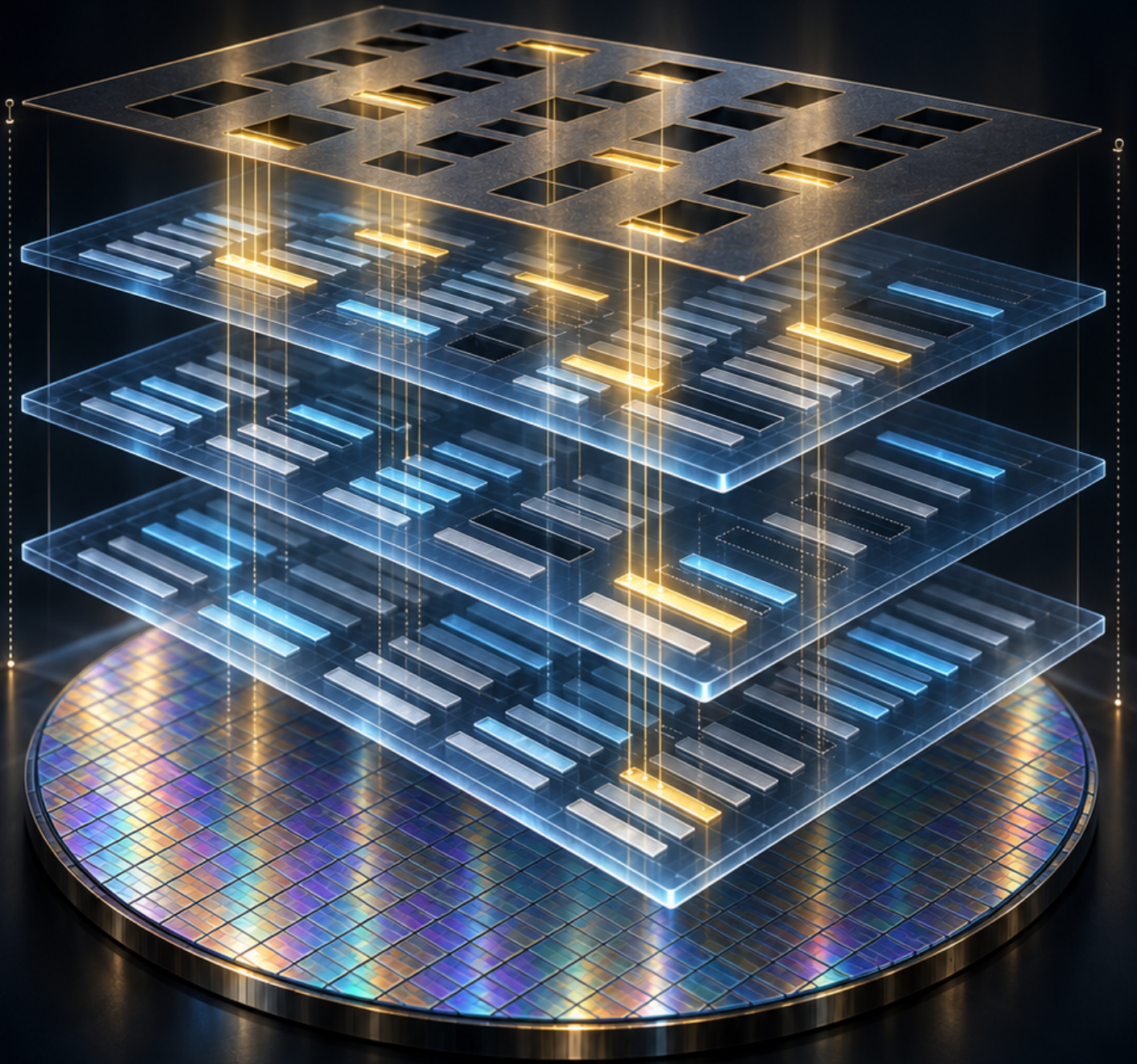


Trace and Log Analysis as a Fabrication Process

Adding and Subtracting Messages
through Patterning



Concept by Dmitry Vostokov

Trace and Log Analysis as a Fabrication Process

Adding and Subtracting Messages through Patterning

Dmitry Vostokov | Software Diagnostics and Observability Institute

Developed with AI assistance (GPT-6 Astra and Claude Fable 5.1)

31 October 2026

Abstract

We propose a fabrication perspective on trace and log analysis in which diagnostic patterns guide the construction of derived execution artifacts. Subtractive operations select messages and remove them from a working representation. Additive operations introduce observations from other artifacts, calculated messages, annotations, or explicitly identified hypotheses. Patterning determines where and under which conditions these operations apply. We connect this perspective with our earlier Software Codiagnosics, Diagnostic Operads, and Higher-Order Pattern Narratives. Codiagnosics supplies the transformation perspective; Diagnostic Operads supplies the composition of diagnostic operations; Higher-Order Pattern Narratives supplies the analysis of their actual application and revision. We distinguish the order of transformations from narrative order, and we distinguish recorded observations from derived or inferred messages. Two worked examples demonstrate retry summarization and the recovery of context lost through premature filtering. A compact operational model specifies occurrence identity, ordering, provenance, and preservation obligations. The contribution is a conceptual and methodological synthesis with checkable examples. It does not establish empirical improvements in diagnostic accuracy or claim that filtering, composition, or provenance originates with this approach.

Keywords: trace analysis; log analysis; pattern-oriented diagnostics; software codiagnosics; diagnostic operads; higher-order pattern narratives; provenance; artifact transformation

1 Introduction

A software trace is often too large, too interleaved, or too sparse to support direct diagnostic reading. We select an activity, combine records from several components, calculate time intervals, collapse repeated messages, and introduce annotations. The artifact we eventually explain may differ considerably from the artifact we first opened. Its diagnostic usefulness depends on the transformations that produced it.

We can understand this work as a fabrication process. Messages and their relationships constitute the material. Analysis patterns determine selections and transformations. A sequence of operations produces a derived trace in which some structures become easier to recognize. The practical question is whether the resulting representation preserves the evidence needed for the diagnostic question.

The word fabrication is used here in its constructive sense. The product is a derived diagnostic artifact with a recorded relationship to its inputs. Captured records retain their identity, calculated statements identify their derivation, and hypotheses retain their status as hypotheses. This distinction is part of the method because an apparently coherent trace can otherwise conceal how much of its content was observed and how much was constructed.

Our earlier Software Codiagnosics already considers artifact transformations through analysis patterns [1]. Diagnostic Operads considers how diagnostic operations can be composed [2]. Higher-Order Pattern Narratives considers the narratives of analysis and their subsequent analysis [3]. We bring these perspectives together through the fabrication analogy and give them operational interpretations for adding and subtracting messages.

We address four questions. What is added or removed when we transform a trace? How do patterns determine these changes? How can operations be composed without losing their diagnostic meaning? How can we

examine the analysis process when a transformation hides relevant evidence? The intended readers are software diagnosticians, developers, support and reliability engineers, and researchers working with execution artifacts.

2 Fabrication and Patterning

Semiconductor fabrication provides a useful source analogy. Thin films are deposited, lithography transfers a pattern into photosensitive material, development opens selected regions, and etching removes exposed material [17]. Processing is repeated as layers are built [4], with measurement and inspection supporting fabrication [17]. In trace analysis, a pattern similarly specifies a region of application, while an operation determines the change made there.

This correspondence separates recognition from action. Recognizing a repeated message block does not specify whether we should retain every occurrence, replace the block with a summary, or compare it with another block. The diagnostic question determines the operation. A repetition count may be sufficient for one question, whereas the spacing between individual occurrences is necessary for another.

Table 1. Proposed correspondence between fabrication and diagnostic analysis.

Fabrication concept	Diagnostic interpretation
Substrate	The preserved source artifacts from which working representations are constructed
Patterning	A selection mask identifying occurrences, intervals, activities, or relationships
Deposition	Addition of imported observations, derived records, or labeled hypotheses
Etching	Removal of selected occurrences from a working representation
Alignment	Correlation of sources using identities, clock information, and activity boundaries
Inspection	Examination of the result and the operations that produced it
Repeated processing	Successive construction and analysis of derived artifacts

The analogy has limits. A trace carries assertions about execution, and an operation changes how those assertions are represented. Removing a message from a view does not remove the event from execution history. Adding an inferred step does not establish that the step occurred. Digital representations also support branching and replay from preserved sources more readily than physical fabrication supports undoing a destructive process.

Patterning may operate at several scales. A message predicate selects individual occurrences. An interval rule selects a region around an anchor. A correlation rule selects an activity distributed across files. A structural rule selects a repeated block or a relationship between blocks. The operation must state its scale because a message-level filter can break a relationship that only becomes visible at the activity level.

The term selection mask will denote a rule governing retention or removal in this paper. Our existing Trace Mask pattern has a different, specific meaning: superposition of traces to construct a new log and impose additional structure [6]. Both fit the fabrication perspective, but their operations should remain distinguishable. A selection mask determines membership in a view; Trace Mask combines material from traces.

3 Earlier Work and Related Foundations

Software Codiagnosics

Software Codiagnosics names the transformation of execution artifacts through analysis patterns so that additional diagnostic indicators become accessible. Its cooperative and duality interpretation also draws

attention to reversible or partly reversible transformations [1]. We use fabrication as an interpretation of this transformation activity. The working artifact becomes something whose construction can be specified and inspected.

The distinction between source and working representation is useful here. If a transformation only hides occurrences, we can recover them by changing the view over the source. If a transformation overwrites the only available copy, that recovery may be impossible. A reference to an omitted record is useful only while its source remains available and identifiable.

Diagnostic Operads and Working Sets

Diagnostic Operads proposes decomposing high-level analysis patterns into elementary operations, with multiple input artifacts and parameters where required. Its preliminary diagnostic meaning is an operational sequence used to obtain diagnostic indicators [2]. Working Set provides an explicit connection: an analyst's current selection of log messages can be constructed by chaining diagnostic operads, in the practical sense of [2], from the original traces [8].

In our fabrication interpretation, an operational sequence is a recipe. A recipe can combine selection from one pattern, correlation from another, and a summary operation from a third. Its interpretation depends on operation order and on the intermediate artifacts available to later operations. Section 6 distinguishes this practical meaning from a stricter mathematical operad construction.

Higher-Order Pattern Narratives and CoTrace

A pattern narrative expresses an execution artifact in terms of recognized patterns. Our earlier account distinguishes this representation from the analysis narrative recording how the patterns were found, revised, or discarded. The analysis narrative has its own Analysis Time and can itself be analyzed [3]. CoTrace supplies a related representation of the messages or regions visited during analysis, in an order that can differ from their execution order [9].

These concepts allow us to inspect the production of a derived artifact. A successful result can obscure an inefficient path, and an unsuccessful result can reveal where a useful observation was discarded. Recording the actual analysis makes both cases available for examination. The interpretation of such a record as a pattern narrative requires identifying patterns in the analysis activity.

Data Transformation and Provenance

Selection, combination, and transformation of data have established foundations. Codd's relational model provides an early account of relations and operations on them [5]. Green, Karvounarakis, and Tannen develop algebraic annotations for data provenance, including polynomial representations of contributions to query results [12]. W3C PROV-DM describes entities, activities, agents, derivation, plans, and provenance of provenance [13].

Our proposal builds on these foundations. A diagnostic fabrication system can use database operations and a standard provenance representation. Its domain-specific content concerns the patterns that motivate a transformation, the diagnostic observations that must survive it, the status of added statements, and the investigation of the analysis itself. An execution history with provenance is therefore an available foundation for higher-order diagnostic interpretation.

4 Messages as Diagnostic Material

Occurrences and Relationships

Two identical text lines can represent different observations. They may record distinct retries, repeated failures, or duplicate delivery of the same event. We consequently work with message occurrences carrying identities. Identity should distinguish a source artifact and a position or stable record identifier within it. Text equality alone is insufficient grounds for deduplication.

For the present model, a trace consists of identified occurrences, message content and metadata, source positions, and ordering information. The metadata includes clock domain, correlation identifiers, and provenance. A single file can supply a sequence order. A collection of distributed traces may supply only a partial order. A deterministic display can extend that order, but its additional ordering choices remain presentation decisions.

Lamport's treatment of distributed events explains why temporal order requires care: causal precedence is a partial order, and clock-based ordering has explicit assumptions [11]. Accordingly, an imported record with an earlier timestamp is not automatically the cause of a later record. Correlation identifiers associate observations; they do not by themselves prove every proposed causal relation between them.

Subtractive Operations

A subtractive operation applies a selection mask to change membership in the working representation. It may retain one activity, remove a known background message class, select an interval, or replace several visible occurrences with a compact representation. In the last case, subtraction is paired with an additive operation that produces the replacement.

Subtraction is relative to a diagnostic question. Heartbeat messages may be incidental when reconstructing a request, yet essential when investigating liveness, scheduling gaps, or message density. Calling them noise therefore records a decision within an analysis context. A recipe should state which question justifies excluding them and which relationships must remain recoverable.

The scope of a selection also matters. Selecting all errors can remove the start and completion messages needed to measure an operation. Selecting one component can remove evidence from a dependency. Selecting a time interval can cut an activity in half. A pattern-oriented selection can retain boundary messages or expand through correlation links when its initial selection is insufficient.

Additive Operations

We distinguish four kinds of additions. Imported observations retain their original source identity. Derived messages express calculations or recognized structures, such as a duration or repetition count. Inferred messages express hypotheses about unrecorded steps. Padding messages support a representation without asserting an execution event. The distinction between the Implicit Message and Silent Messages patterns already separates inference from padding in the pattern catalogue [7, 15].

Table 2. Meaning and provenance of additions to a working trace.

Addition	Example	Required interpretation
Imported observation	A server error added to a client view	An existing record from an identified source
Derived message	Two retries observed between 10 ms and 20 ms	A calculation with contributing record identities and a stated rule
Inferred message	A resource acquisition may have preceded a failure	A hypothesis with supporting evidence and assumptions
Padding message [15]	An empty slot inserted on a regular time grid	A representational device carrying no claim that execution occurred

An annotation need not be inserted as an apparent event between two observations. It can be attached to an interval or a set of contributing records. This avoids assigning an invented event timestamp to a summary. The calculation time belongs to the analysis history; the summarized interval belongs to the execution representation.

Our Implicit Message example concerns a possible intermediate operation between an intention to read a file and an access-denied outcome [7]. In applying that idea, we preserve alternatives: the failure could arise in a library, policy check, or remote service. The inferred statement must be no more specific than the available evidence supports. A useful hypothesis can remain useful while its implementation details are unresolved.

5 A Compact Operational Model

We now make the construction explicit. A recipe step uses a working view and, when declared, preserved source artifacts to produce a new view. Its pattern specification and parameters determine the operation. The source collection is read-only within this model. A materialized output is a new artifact rather than a replacement of its sources.

Each working view records the identities it retains, the records it adds, and the relationships used to present them. Added records have fresh identities, a kind from Table 2, and derivation information. Reusing an added identity for changed content is disallowed. If a result is revised, the revision receives a new identity or an explicitly versioned identity.

Selection and Derivation

Let T be the input working view. For a simple recipe, let Sel_A select Request A , let G add a retry summary, and let R remove the retry occurrences from the visible view. Read function composition from right to left:

$$F = R \circ G \circ \text{Sel}_A$$

$$F(T) = R(G(\text{Sel}_A(T)))$$

G must declare whether it reads its current input or the preserved source. In this example it reads the current input. R preserves the derived summary and its source references while removing the individual retry occurrences from display.

The parameters of G include the grouping key, the event definition being counted, and the treatment of repeated deliveries. The recipe also declares the domain of the calculation: observed retry messages, rather than all retries that may have occurred. This small qualification prevents an accurate count of recorded occurrences from becoming an unsupported statement about complete execution.

Preservation Obligations

A transformation can preserve one observation while losing another. Let Q be a diagnostic query that is defined on both the original and the derived representation. A recipe may state the following obligation for a declared class of inputs:

$$Q(F(T)) = Q(T)$$

For example, Q can ask for the number of observed retries for Request A. On the source it counts the relevant occurrences; on the derived representation it reads a verified summary. Equality requires compatible query meanings and adequate provenance. It does not follow from the fact that both outputs contain the word `retry`.

We cannot assume the same equality for every query. A count-only summary may preserve the number of occurrences while losing their spacing. A selection of one request may preserve that request's recorded duration while changing global message density. A recipe therefore states its preservation obligations alongside known losses. A smaller artifact is not, by itself, evidence of a better diagnosis.

Ordering and Noncommutativity

In general, the order of operations changes the result. If R removes every retry occurrence before G counts them, G observes zero retries. If G executes first, it observes the original count. The two compositions can therefore disagree:

$$R(G(T)) \neq G(R(T))$$

This example concerns operations reading their immediate inputs. A source-reading summary can still compute the original count after filtering, but that is a different operation with a different input dependency. Making these dependencies explicit prevents an apparent contradiction and allows tools to check whether a proposed reordering is valid.

Reversibility and Information Loss

A retained source and an omission manifest allow reconstruction of a previous view by replay or reselection. This is recoverability through additional information. A summary alone normally has no inverse: many different retry sequences can have the same count and interval. Addition and subtraction of messages consequently do not establish a group structure or arithmetic cancellation law.

Source preservation also has a practical boundary. Replay needs the source bytes, the decoding rules, the pattern version, and the parameters used. A hash can help identify a source version, but a hash does not recover missing content. Any claim of reproducibility must state which of these inputs remain available.

6 Diagnostic Operads as Fabrication Recipes

The practical diagnostic notion of an operad gives us a way to assemble operations that come from different patterns [2]. A request reconstruction might combine a correlation operation, an interval selection, a recognition step, and a calculated message. Several inputs are natural: joining a client trace with a server trace cannot be expressed as an operation on only one unaugmented input.

Mathematical operads concern operations and their composition [10]. For a precise realization of our proposal, we can assign types to inputs and outputs, such as decoded trace, correlated trace, interval selection, and annotated view. An operation accepts its declared input types and returns its declared output type. A compatible output can be substituted into an input position of another operation.

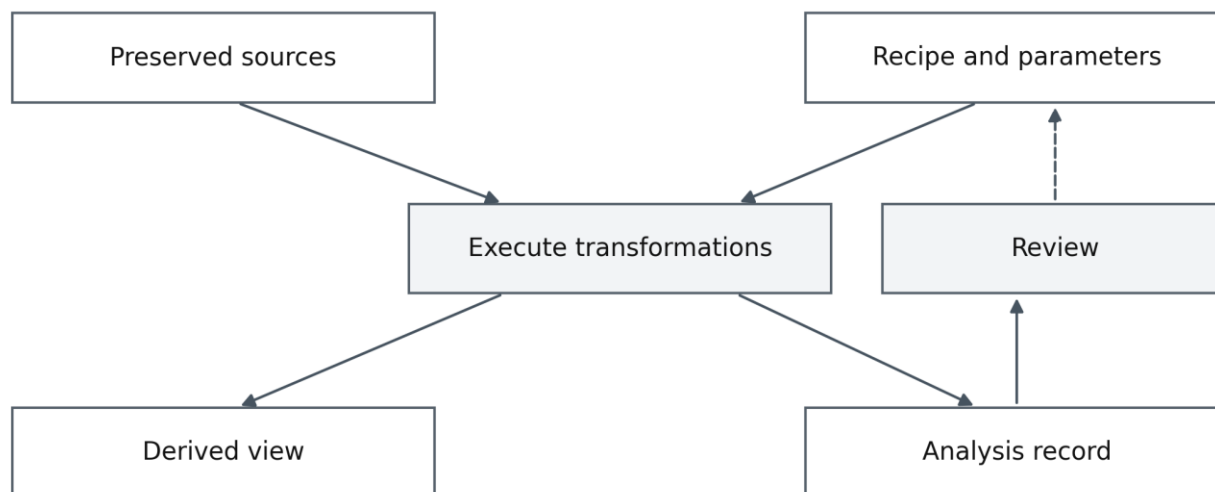
Consider a binary correlation operation J , two source-specific selection operations U and W , and a summary operation G . The composite expression $G(J(U(C), W(D)))$ specifies a recipe over client source C and

dependency source D. It also exposes two places where source-specific filtering can destroy evidence before correlation. A single linear command transcript can obscure this branching dependency.

In the strict mathematical interpretation, a recipe is a composite operation. With ordered input slots and no chosen permutation action, a typed collection closed under substitution and containing identities forms a nonsymmetric colored operad, equivalently a plain multicategory, when substitution satisfies associativity and unit laws [16]. We retain the earlier diagnostic terminology while making this distinction explicit. Assigning an operadic name to an arbitrary workflow does not establish the required mathematical laws.

One tractable realization assigns a set of artifact representations to each type and uses deterministic, side-effect-free functions from finite ordered products of those sets to an output set. The full collection is the colored endomorphism operad of this family, considered without its permutation action; a chosen subcollection must contain identities and be closed under substitution. Functional substitution supplies the laws: regrouping substitutions preserves the function, and inserting identities changes nothing. This associativity does not permit exchanging filtering and summarization. Failures must be represented explicitly, for example by result types and failure-aware functions, so every operation is total on its declared inputs.

Interactive investigations add observations and change parameters during execution. Their realized histories also contain timestamps, tool failures, and analyst decisions. We therefore distinguish the semantic recipe from its particular execution record. An identity transformation can leave the artifact unchanged even when observing its execution adds a record to an external analysis history.



Source references connect derived records to their contributors

Figure 1. Sources and a recipe produce a derived view and an analysis record. Review of the recorded investigation can lead to a revised recipe. The sources remain available for replay.

7 Worked Examples

The following examples are constructed fixtures. Their records, timestamps, and identifiers are specified in full so that the transformations can be checked directly. They illustrate the proposed method and do not report observations from a production incident. All times are relative milliseconds on a shared fixture clock; no distributed clock synchronization is assumed outside these examples.

Retry Summarization

Consider the six occurrences in Table 3. The diagnostic question is how many retries were recorded for Request A before its timeout, and how long the recorded request interval lasted. The two heartbeat messages belong to independent activity B.

Table 3. Input trace for the retry example.

ID	Time ms	Activity	Message
m1	0	A	Request started
m2	5	B	Heartbeat
m3	10	A	Retry
m4	15	B	Heartbeat
m5	20	A	Retry
m6	30	A	Request timed out

Selection by activity A retains m1, m3, m5, and m6. The derivation operation counts m3 and m5 and creates d1 with the content Two observed retries. Its support is the pair of source occurrences m3 and m5; its execution interval runs from 10 to 20 ms. Its analysis creation time is recorded separately. The subtraction operation then removes m3 and m5 from the visible representation.

Table 4. Derived working representation for Request A.

Record	Placement	Meaning	Source support
m1	0 ms	Observed request start	m1
d1	Interval 10 to 20 ms	Derived summary of two observed retries	m3 and m5
m6	30 ms	Observed request timeout	m6

The final representation has three visible records. Two are observations, and one is a derived statement. The count is two; the recorded request interval is 30 ms. The summary also retains its two contributing identities, so the original retry timestamps can be recovered while the source remains available. This result preserves the declared count and interval queries.

Several other properties need separate treatment. A summary without source access would lose the individual spacing of the retry occurrences. The original trace contains six observations, whereas the derived display contains three records of mixed kinds. Computing an event rate from the number of displayed rows would therefore change both the selected population and the counting unit. Display compression is not an event-rate measurement.

Reversing the critical operations demonstrates a failure. Removing the two retry occurrences before applying a current-input counter yields zero visible retries. That value describes the filtered input to the counter, but it fails the recipe's preservation obligation about the original observations. A tool can reject the recipe or report the lost dependency instead of emitting an apparently equivalent summary.

Reapplying the summarizer also requires a policy. It can add another summary and cause double counting if later operations treat both summaries as independent evidence. A derivation key based on source identities, pattern version, and parameters can identify an existing result. Alternatively, a new result can explicitly supersede the earlier summary. Idempotence must be designed or checked for the chosen operation.

Recovering Context Lost by Component Filtering

The second example contains records from components A and B. Request identity R is propagated between them, while X denotes unrelated activity. Table 5 presents the combined fixture order.

Table 5. Two components contributing to one request.

ID	Time ms	Component	Request	Message
a1	0	A	R	Request started
b1	5	B	R	Dependency request received
b2	8	B	R	Request rejected by policy
a2	10	A	R	Retry
h1	12	B	X	Heartbeat
a3	20	A	R	Retry
a4	30	A	R	Request timed out

An initial recipe selects component A. Its result contains a1, a2, a3, and a4. The analyst recognizes a timeout preceded by retries and can construct a correct local summary. The component selection nevertheless excludes the recorded rejection in b2. The local summary does not answer why the dependency interaction failed.

A revised recipe selects request R across both sources. It imports b1 and b2 into the working representation while retaining their identities and component attribution. It can then add a derived statement: Component B recorded a policy rejection before the recorded retries and timeout in A. This statement is supported by the fixture's identifiers and clock assumptions. It does not establish the complete root cause of the policy decision or prove that every retry encountered the same rejection.

The new recipe addresses a different preservation obligation: retain observations associated with the request across component boundaries. A rule that only preserves local retry count is insufficient for that purpose. The change in diagnostic question explains the change in construction rather than making the original local calculation arithmetically wrong.

The rejected component restriction also belongs in the analysis record. Without it, the final report shows the broader context but gives no indication that the investigation initially concealed it. That history supports analysis of the diagnostic process, as developed next.

8 Higher-Order Pattern Narratives of Fabrication

The history of fabrication records what happened during the investigation: selections attempted, patterns recognized, summaries added, results reconsidered, and evidence restored. It has a chronology of its own. A timeout may be the first pattern recognized even though the corresponding event occurred near the end of the execution trace.

We preserve the narrative-order distinction in our earlier account [3]. An execution narrative and its pattern representation occupy the same order. The narrative of analyzing them occupies the next order. A review of that analysis can produce a further order. Several transformations of one execution artifact do not, by themselves, raise its narrative order.

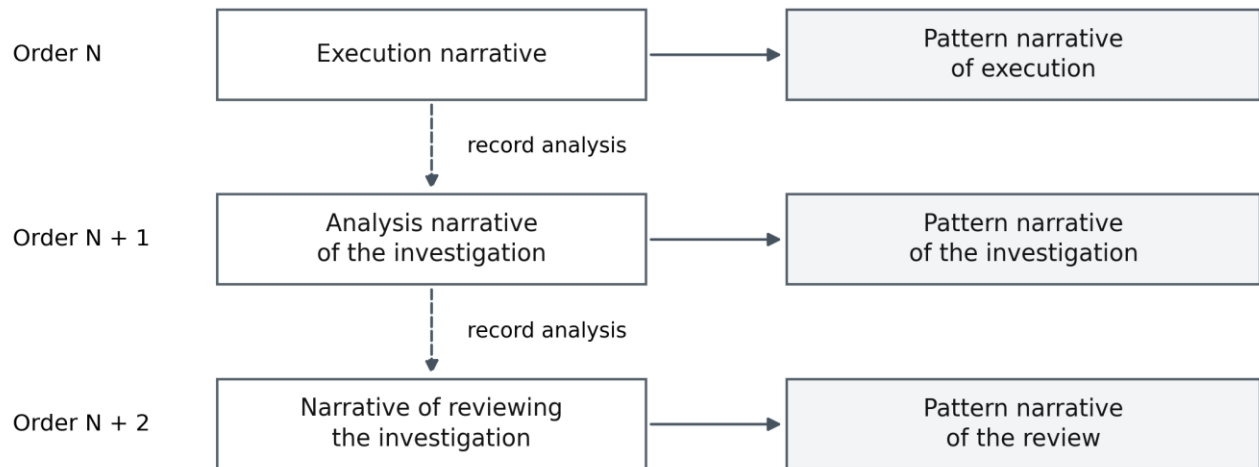


Figure 2. Pattern representation and narrative order are different dimensions. Horizontal arrows construct pattern representations at the same order. Vertical arrows record the analysis of the preceding order.

A Recorded Investigation

For the second example, an analysis narrative could record the sequence in Table 6. Analysis Time is the ordering coordinate of the investigation; it must not be confused with the fixture's execution timestamps.

Table 6. Analysis narrative for the component-filtering example.

Analysis step	Action and result
TA1	Select component A and recognize Timeout at a4
TA2	Recognize the retry occurrences a2 and a3 and add a summary
TA3	Inspect request identity R and search the preserved component B source
TA4	Find the policy rejection b2 and revise the component restriction
TA5	Rebuild the request view and qualify the earlier local interpretation

TA1 through TA5 specify order but not elapsed analysis durations. Measuring how long the analyst took requires actual timestamps or durations. This distinction prevents an illustrative history from being treated as performance evidence. A real implementation should capture both ordered steps and the clock information needed for timing claims.

The artifact at the end of TA5 may be the same artifact another analyst constructs immediately. Their analysis narratives still differ. One history includes restoration and revision; the other may follow a broader initial correlation rule. Comparing such histories can identify a useful difference in procedure, subject to differences in prior knowledge, tools, and available evidence.

Diagnosing the Analysis

A review of Table 6 can identify premature restriction to one component as the step that excluded relevant context. The reviewer can propose an inspection after component selection: determine whether the request identity appears in other sources before accepting a complete explanation. This is a diagnostic observation about the recipe's use in this case.

An operational log is a useful starting point, but a pattern narrative adds interpretation. A list of queries records actions. Recognizing repeated abandonment of the same filter, a discontinuity in recorded reasoning, or an extended interval before cross-source correlation describes patterns in the analysis activity. Such observations require adequate records; silence in the operation log may reflect missing instrumentation rather than an analyst's inactivity.

CoTrace provides a practical bridge by recording the occurrences or regions visited during an investigation [9]. A fabrication record can extend that information with operation parameters, view versions, additions, omissions, and recorded reasons. Diagram was also proposed for presenting and analyzing higher-order pattern narratives [14]. The diagrams in this paper are schematic representations of the present model.

Higher-order inspection should have a defined purpose. Reviewing an analysis is useful when it explains an incorrect conclusion, unnecessary work, or an unstable procedure. Repeatedly generating higher orders without a diagnostic question merely produces more material. The same selection and preservation questions apply when the analysis history becomes the next artifact.

9 Provenance and Inspection

The construction history should permit a reader or tool to answer concrete questions. Which source version supplied an observation? Which records contributed to a summary? Which rule excluded an occurrence? Which assumptions support an inferred message? Which result was later revised? These questions concern relationships between records, operations, and decisions.

A minimal operation record contains the operation identity and version, input artifact versions, pattern and parameter identifiers, output artifact version, and analysis ordering information. For an additive operation it also records new identities and their kinds. For a subtractive operation it records either the omitted identities or an exact reproducible selection mask over immutable inputs. A statement such as noise removed is insufficient to reconstruct the decision.

Lineage is necessary but does not establish semantic correctness. Two source identities can be correctly attached to a duration calculated in incompatible clock domains. An inferred message can cite genuine records while making an unsupported interpretation. Inspection therefore checks both the existence of support and the validity of the rule that uses it.

W3C PROV-DM can represent artifacts as entities, executions as activities, and responsible people or software as agents. It also supports plans and provenance of provenance [13]. A diagnostic implementation can attach domain attributes for pattern identity, observation kind, and preservation obligations. We do not need to invent a replacement for general provenance to record the proposed fabrication process.

The analogy also suggests inspection points between operations. Before suppressing repetitions, verify that a summary refers to the intended occurrences. Before joining traces, verify identity scope and clock assumptions. Before accepting a derived explanation, check that it does not silently depend on a removed boundary record. These checks are chosen because they protect a stated diagnostic requirement.

When an upstream artifact changes, downstream summaries and hypotheses require reconsideration. An earlier summary can remain a valid statement about its original input version while being obsolete for the new view. A dependency record makes that distinction explicit and prevents a current report from accidentally combining observations and summaries from incompatible versions.

10 Novelty and Contribution

The proposed contribution is a synthesis of established operations and our earlier diagnostic concepts. We claim a particular organization and application of these ideas. The paper does not establish worldwide priority

for using a manufacturing metaphor in data analysis, and it does not report a systematic review of every related workflow system.

First, we formulate additive and subtractive patterning as complementary construction operations on diagnostic artifacts. The pattern identifies a region or relationship; the chosen operation changes the working representation. This separation makes it possible to ask why a recognized structure was summarized, retained, or removed in a particular investigation.

Second, we connect the transformation, composition, and analysis-history perspectives in one account. Software Codiagnosics explains the role of artifact transformation. Diagnostic Operads explains the composition of diagnostic operations. Higher-Order Pattern Narratives explains how their application and revision becomes diagnostic material. The integration makes the fabrication recipe and its realized investigation separately inspectable.

Third, we give the analogy operational obligations. Occurrence identity protects multiplicity; typed additions distinguish observation from derivation and inference; declared dependencies govern operation order; and preservation obligations connect a transformation to its diagnostic purpose. These mechanisms are familiar in other settings. Their proposed combination here explains how a reduced trace can remain adequate for one question while becoming inadequate for another.

Fourth, we make narrative order independent of transformation depth. Filtering, joining, and summarizing can produce several versions at one narrative order. Recording the investigation produces the next order, and analyzing that investigation can produce another. This distinction prevents a succession of processed files from being mistaken for a succession of higher-order narratives.

Table 7. Established foundations and the proposed development.

Foundation	Established contribution	Development in this paper
Relational operations [5]	Operations over structured data	Pattern-guided selection tied to diagnostic preservation obligations
Provenance [12, 13]	Contributions, derivations, plans, and execution histories	Explicit provenance requirements for additions, omissions, and diagnostic revisions
Software Codiagnosics [1]	Artifact transformation through analysis patterns	Additive and subtractive fabrication interpretation
Diagnostic Operads [2]	Composition of diagnostic operations	Recipes with declared input dependencies and inspection points
Higher-Order Pattern Narratives [3]	Analysis narratives and their further analysis	Examination of fabrication choices, losses, and repairs

The examples support limited claims. They establish the stated counts for fully specified inputs, show a concrete order-sensitive failure, and demonstrate how component filtering can exclude relevant records. They do not establish that the approach improves human performance on large traces. That claim requires a separate evaluation.

11 Limitations and Evaluation

The first limitation is the adequacy of the patterns and questions. A reproducible recipe can consistently make an inappropriate selection. A correct summary of selected evidence can coexist with an incomplete explanation. Source preservation makes review possible, but it does not ensure that the analyst recognizes which omitted records matter.

The second limitation is incomplete capture. A log may omit events before analysis begins. A rule reporting no retry messages in a view cannot conclude that no retries occurred unless the relevant completeness assumptions are established. Hypothesis insertion can assist reasoning about gaps, but it cannot repair the evidential status of the original capture.

The third limitation concerns costs. Occurrence-level lineage and analysis histories consume space, and repeated reconstruction consumes time. Large implementations may need indexed selections, compact omission descriptions, checkpoints, and retention policies. Their correctness depends on preserving enough information for the declared replay and inspection requirements.

The fourth limitation concerns human and automated decisions. An analyst may not record every reason for a transformation. An automated system may produce plausible annotations without sufficient support. The same requirement applies to both: an added message should expose its evidential kind and derivation. If a language model proposes an operation or hypothesis, its output enters the analysis as a proposal that can be tested against the available artifacts.

A practical evaluation would compare investigations using the same source artifacts and diagnostic questions. Measures could include incorrect conclusions, successful recovery of hidden context, time to a supported explanation, agreement between reviewers, and completeness of replay. A smaller display should be measured separately from diagnostic success. Otherwise a method that removes crucial context could appear successful merely because it reduces the number of rows.

Evaluation should also separate the contributions of recipe recording, provenance, and higher-order review. If a complete system outperforms a baseline, that result alone does not identify which part helped. Controlled comparisons can examine whether adding analysis narratives improves recovery from a deliberately introduced filtering error, and whether retaining source references improves verification of summaries.

At the operation level, useful checks follow directly from declared obligations. Applying a selection mask can be checked for preservation of source-relative order. A summary can be checked against its contributors. Reordering can be attempted only where equivalence has been established for the relevant input class. The two fixtures in Section 7 supply small cases for these checks; they are not a substitute for validation on representative diagnostic work.

12 Conclusion

Trace and log analysis can be described as the construction of diagnostic artifacts through additive and subtractive patterning. The method becomes explicit when each transformation declares its inputs, its additions and omissions, the questions it preserves, and the information needed to inspect its result.

The fabrication perspective develops Software Codiagnosics through the material interpretation of artifact transformation. Diagnostic Operads organizes the operations into recipes. Higher-Order Pattern Narratives makes their actual application and revision available for further diagnosis. Together, these perspectives let us examine both the patterns exposed in a constructed trace and the analysis decisions that made them visible.

The next practical step is to represent these decisions in analysis tools and evaluate them on investigations where an apparently useful reduction can conceal relevant context. The two worked examples identify the essential requirement: the diagnostic meaning of a derived trace must remain connected to the observations and operations from which it was constructed.

References

Web references were accessed on 29 October 2026.

- [1] Vostokov, D. Software Codiagnosics. Software Diagnostics and Observability Institute. <https://dumpanalysis.org/codiagnosics>
- [2] Vostokov, D. Diagnostic Operads. Software Diagnostics and Observability Institute. <https://www.dumpanalysis.org/diagnostic-operads>
- [3] Vostokov, D. Higher-Order Pattern Narratives (Analysing Diagnostic Analysis). Software Diagnostics and Observability Institute. <https://www.dumpanalysis.org/pattern-narratives>
- [4] ASML. How microchips are made. <https://www.asml.com/en/technology/all-about-microchips/how-microchips-are-made>
- [5] Codd, E. F. A relational model of data for large shared data banks. Communications of the ACM, 13(6), 377–387, 1970. <https://doi.org/10.1145/362384.362685>
- [6] Vostokov, D. Trace Analysis Patterns (Part 104): Trace Mask. Software Diagnostics Library, 17 March 2015. <https://www.dumpanalysis.org/blog/index.php/2015/03/17/trace-analysis-patterns-part-104/>
- [7] Vostokov, D. Trace Analysis Patterns (Part 249): Implicit Message. Software Diagnostics Library, 16 March 2025. <https://www.dumpanalysis.org/blog/index.php/2025/03/16/trace-analysis-patterns-part-249/>
- [8] Vostokov, D. Trace Analysis Patterns (Part 155): Working Set. Software Diagnostics Library, 26 November 2017. <https://www.dumpanalysis.org/blog/index.php/2017/11/26/trace-analysis-patterns-part-155/>
- [9] Vostokov, D. Trace Analysis Patterns (Part 175): CoTrace (CoLog, CoData). Software Diagnostics Library, 21 July 2019. <https://www.dumpanalysis.org/blog/index.php/2019/07/21/trace-analysis-patterns-part-175/>
- [10] Loday, J.-L., and Vallette, B. Algebraic Operads. Grundlehren der mathematischen Wissenschaften, volume 346. Springer, 2012. <https://doi.org/10.1007/978-3-642-30362-3>
- [11] Lamport, L. Time, clocks, and the ordering of events in a distributed system. Communications of the ACM, 21(7), 558–565, 1978. <https://doi.org/10.1145/359545.359563>
- [12] Green, T. J., Karvounarakis, G., and Tannen, V. Provenance semirings. Proceedings of the 26th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, 31–40, 2007. <https://doi.org/10.1145/1265530.1265535>
- [13] Moreau, L., and Missier, P., editors. PROV-DM: The PROV Data Model. W3C Recommendation, 30 April 2013. <https://www.w3.org/TR/2013/REC-prov-dm-20130430/>
- [14] Vostokov, D. Diagram Graphical Diagnostic Analysis Language. Software Diagnostics and Observability Institute. <https://dumpanalysis.org/diagram-diagnostic-analysis-language>
- [15] Vostokov, D. Trace Analysis Patterns (Part 79): Silent Messages. Software Diagnostics Library, 20 April 2014. <https://www.dumpanalysis.org/blog/index.php/2014/04/20/trace-analysis-patterns-part-79/>
- [16] Leinster, T. Higher Operads, Higher Categories. London Mathematical Society Lecture Note Series, volume 298. Cambridge University Press, 2004, Chapter 2. Author's preprint: <https://arxiv.org/abs/math/0305049>
- [17] Li, A. Six crucial steps in semiconductor manufacturing. ASML, updated 4 October 2023. <https://www.asml.com/en/company/stories/2021/semiconductor-manufacturing-process-steps>