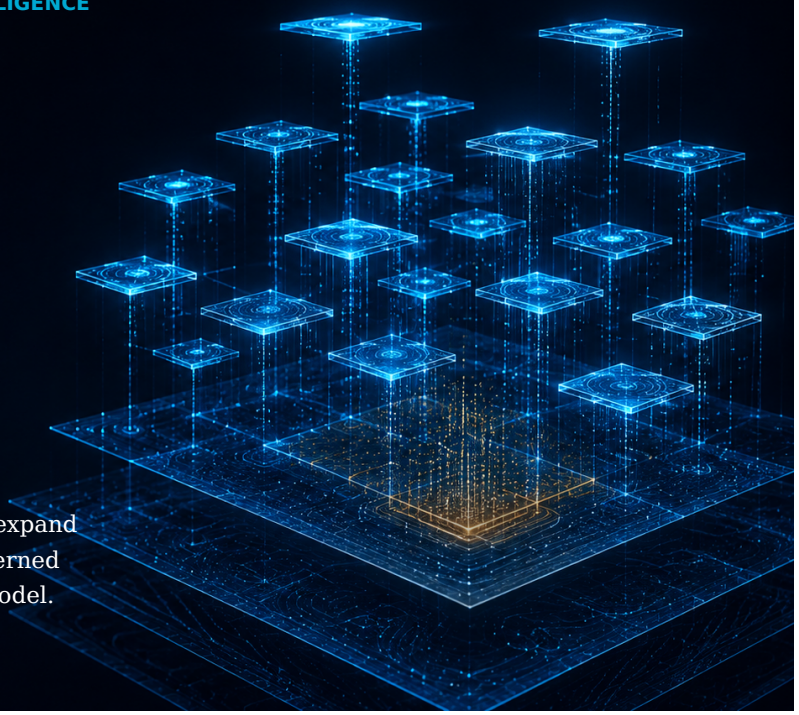
WHITE PAPER 08 / GOVERNED SCIENTIFIC INTELLIGENCE

Micro-Engines, Not a Monolith



Bounded analytical components let BENNETT expand scientific capability without rebuilding the governed foundation or concentrating authority in one model.

DR. BRIAN T. BENNETT
Co-Founder and CSO

PAPER 08 OF 10
AUGUST 29, 2026

NATIVE EVIDENCE / DETERMINISTIC MEASUREMENT / GOVERNED INTERPRETATION / DEFENSIBLE PUBLICATION

CONTENTS

Paper structure

Bounded analytical components let BENNETT expand scientific capability without rebuilding the governed foundation or concentrating authority in one model.

All numbered citations link to the corresponding reference. Linked source identifiers in the reference list open the cited source.

ABSTRACT

Paper 7 ended with a requirement that is architectural rather than merely user-facing. Scientist-in-Control AI allows a qualified expert to move between guided operation and direct scientific control while preserved evidence, analytical state, provenance, validation obligations, and claim boundaries remain continuous. That requirement exposes a fundamental limitation of trying to make one general-purpose language model carry the entire scientific workflow. Eligibility, measurement, analytical state, validation, interpretation, exception handling, and downstream consequence are different scientific authorities. Collapsing them into one opaque conversational layer may produce fluent answers, but it does not produce a governable scientific system. A governed architecture must instead be able to identify what each component is responsible for, what evidence it requires, what state it may change, what downstream work depends on it, and what must happen when that component is unavailable, invalid, stale, or superseded.[1]

A single undifferentiated analytical monolith is a poor fit for that requirement. When one model, workflow, or opaque analytical block is responsible simultaneously for applicability, measurement, state change, validation, exception handling, interpretation, and downstream routing, scientific responsibility becomes difficult to localize. A successful output can hide which prerequisite authorized the operation. A change can make it unclear which downstream result is stale. A failure can become a generic system error rather than a scientifically meaningful blocked route. A new capability can expand the system in ways that are difficult to test without retesting everything around it.

BENNETT takes a different approach. Scientific work is decomposed into bounded analytical responsibilities. Parent engines organize coherent scientific domains. Micro-engines perform narrower, testable analytical responsibilities within those domains. Each component operates within an explicit scientific scope: it receives defined evidence and state, applies a bounded method or decision rule, produces defined outputs and state consequences, preserves provenance, exposes negative paths, and remains subject to validation and claim boundaries. Components become useful together because their handoffs are explicit rather than because their internal responsibilities are blurred.

The value of this architecture is not decomposition for its own sake. It is **controlled extensibility**. A new scientifically bounded analytical capability can be added, tested, versioned, governed, and connected to the existing platform without redefining the authority of the native evidence, the complete experiment, scientist judgment, validation, or claim boundaries. The supportable capability surface can grow while the governing scientific foundation remains stable.[1,2]

This paper explains that architectural logic at a deliberately bounded level—enough to evaluate the scientific and governance principles without disclosing BENNETT's proprietary implementation details. It describes engines and micro-engines as units of scientific responsibility; evidence-aware applicability; dependency-aware composition; state propagation; local invalidation and revalidation; negative paths; versioning; controlled expansion; the intentionally narrow role of the language model; and the relationship between

component execution and scientist control. It deliberately does **not** publish BENNETT's proprietary dependency maps, registry schemas, transition payloads, exact thresholds, ranking logic, validation fixtures, test matrices, source code, or engineer-ready implementation contracts.[8]

> Central thesis: BENNETT scales scientific capability by composing bounded analytical components whose evidence requirements, applicability, state consequences, validation obligations, provenance, and claim limits are explicit. A micro-engine adds capability only within its supported and validated scope; it inherits the governed scientific foundation rather than redefining it.

1. Paper 7 ended with an expansion requirement

Paper 7 established Scientist-in-Control AI as a governed operating model over preserved evidence. The native acquisition record and current scientific state define which analytical capabilities are supportable. The qualified expert chooses the scientific objective within that space, can move into direct operation when legitimate judgment is required, and can return to the guided route without losing the evidence trail.[1]

That paper deliberately stopped before explaining the architecture required to make those responsibilities scalable.

The implementation problem is demanding. A governed route must be able to:

- expose only scientifically supportable or conditionally supportable capabilities;
- execute deterministic analytical work under explicit prerequisites;
- pause at legitimate scientist-decision boundaries;
- preserve direct expert actions as scientific state;
- identify which outputs remain current after a change;
- invalidate only work that actually depends on a changed state;
- rerun or revalidate affected work when required;
- retain negative paths, holds, blocks, and unsupported outcomes;
- preserve provenance across each handoff; and
- add new capability without allowing that capability to redefine the scientific authority of the system.[1,2]

Paper 7 therefore handed Paper 8 a precise question:

> What implementation architecture allows governed scientific routes to remain composable, testable, dependency-aware, amendable, and expandable as analytical capabilities grow—without rebuilding the governing foundation or concentrating scientific authority in one monolithic model or workflow?

Paper 8 answers that question through **Micro-Engines, Not a Monolith**.

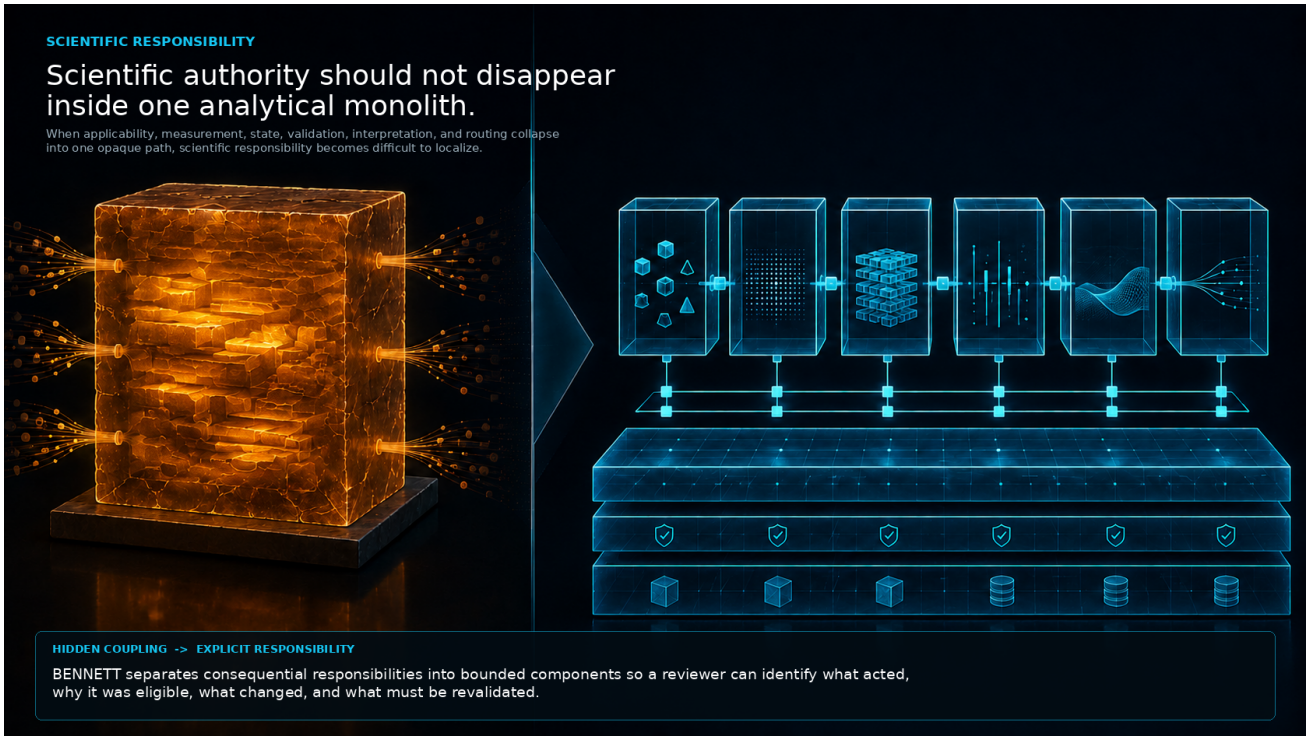


Figure 1. One scientific route should not concentrate every kind of authority in one analytical monolith. Bounded responsibilities separate applicability, measurement, state, validation, interpretation, and routing so scientific authority can remain local and reviewable.

2. A monolith hides where scientific responsibility lives

The word *monolith* can be misunderstood. This paper is not arguing that large software systems are inherently unreliable, that foundation models have no role in science, or that every analytical function must be implemented as a tiny service.

The scientific concern is narrower.

A monolithic scientific workflow concentrates too many consequential responsibilities in one place. If the same opaque system determines whether an analysis is applicable, chooses a method, performs a measurement, interprets a result, decides whether a validation obligation has been satisfied, handles a missing prerequisite, updates state, and constructs the next

route, then a reviewer may see the final output without being able to identify which part of the system carried which scientific authority.

That is incompatible with the evidence model established in Papers 1–7.

Paper 2 defined the complete experiment as a connected evidence-bearing object rather than an image detached from its conditions.[4] Paper 3 established that a method is competent only where the evidence environment supports it.[6] Paper 4 established that material competence is application-dependent.[7] Paper 5 made the native acquisition record authoritative for what the instrument recorded without elevating it to absolute biological truth.[3] Paper 6 separated evidence, measurement, validation, scientist judgment, and model fluency.[2] Paper 7 then showed why expert operation must remain inside that separation of authority.[1]

A scientific architecture should preserve those distinctions in implementation.

The question is not simply whether a computation returns a value. It is:

- What evidence authorized the computation?
- Which component performed it?
- What method and version were active?
- What state did it consume?
- What state did it change?
- Which downstream outputs depend on that state?
- What validation obligation applies?
- What limitations or negative paths remain?
- What part of the final claim can that component legitimately support?

If those questions cannot be answered locally, scientific responsibility becomes difficult to audit globally.

3. Engines organize responsibility; micro-engines bound it further

BENNETT's engineering architecture uses **engines** and **micro-engines** to make scientific responsibility addressable.[5,8]

Generic AI capabilities are often described with labels such as tools, functions, or skills.[9,10] BENNETT uses **engine** and **micro-engine** deliberately. The distinction is not branding for a smaller tool. It denotes a unit of scientific responsibility whose evidence requirements, applicability, state consequences, provenance, validation obligations, negative paths, dependencies, and claim limits are part of the governed architecture.

For purposes of this paper, the terms are described functionally rather than as an enabling software specification.

An **engine** organizes a coherent scientific responsibility or analytical domain. It may concern native-data ingestion, foundational measurement, object or site analysis, temporal analysis, compartmental analysis, modality-aware refinement, localization analysis, assay-level analysis, or another defined responsibility within the platform.

A **micro-engine** is a narrower unit of analytical responsibility within that domain. It is bounded enough that the scientific conditions around its operation can be stated and tested. The relevant questions include:

- What evidence must be present?
- What scientific state must already be accepted?
- What operation is being performed?
- What output or state transition may result?
- What condition limits or blocks the operation?
- What must remain unchanged?
- What provenance must be preserved?
- What downstream work may depend on the result?
- What validation obligation applies before the result can carry further authority?

This is not merely a software-engineering preference for smaller functions. The boundary is scientific.

A micro-engine should be narrow enough that its responsibility can be understood without silently inheriting unrelated authority. A component that measures a spatial quantity should not also become the source of channel identity, material competence, modality validity, or biological mechanism unless those responsibilities are explicitly part of its governed scope.

The same principle prevents an analytical component from becoming more authoritative simply because it is reused often. Reuse demonstrates utility. Scientific authority still comes from the evidence, method, validation, scope, and claim boundary attached to the use.[2]

4. A component boundary is a scientific boundary

Conventional modular software design often emphasizes maintainability, code reuse, deployment, or fault isolation. Those benefits matter, but they are not enough to define a scientific micro-engine.

A BENNETT component boundary exists where a scientific responsibility can be made explicit.

Consider a distance measurement. The arithmetic may be straightforward. The scientific operation is not.

A physical distance requires a resolved source, a scientifically meaningful scope, the relevant

dimensions, valid physical calibration, defined analytical entities, a method appropriate to the evidence state, and units that remain attached to the result. A visually apparent separation in a projection does not automatically authorize a three-dimensional distance. A pixel count does not become micrometers because a user expects a physical number. A distance between two display colors does not establish biological interaction.[3,6]

A bounded component therefore has at least four scientific responsibilities:

1. **Evidence responsibility** — identify the evidence and accepted state required for the operation.
2. **Analytical responsibility** — perform one defined operation without silently absorbing unrelated scientific decisions.
3. **State responsibility** — declare which scientific state the result creates, changes, preserves, or invalidates.
4. **Claim responsibility** — keep the output inside the observational or measurement meaning the method can support.

The boundary makes the component testable because it gives the test something specific to test.

A generic question such as “Does the AI work?” is too broad to govern a scientific platform. A bounded question such as “Under a source with resolved physical calibration and accepted 3D object definitions, does this component return the defined distance, preserve units and provenance, and block unsupported dimensional states?” can be tested.

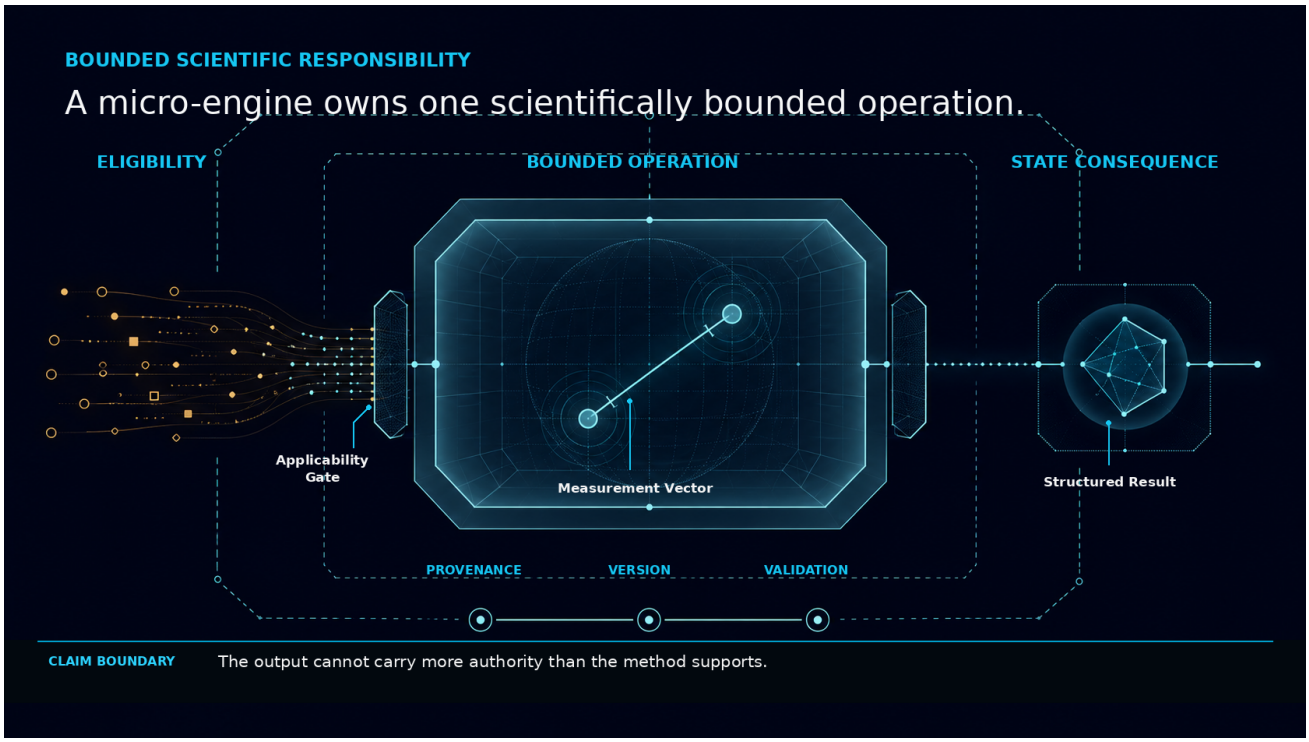


Figure 2. A micro-engine owns one scientifically bounded operation. Evidence and accepted state authorize the operation; provenance, version, validation, and claim limits remain attached to the structured result.

5. Inputs and outputs carry scientific state, not just data

Composition fails if components exchange only numbers while the scientific meaning of those numbers is lost.

Paper 2 established that the experiment is a graph of dependencies.[4] Paper 5 showed that normalized representations remain accountable to the native source and that derivatives require parent lineage.[3] Paper 6 made scientific state explicit.[2] Paper 7 showed that expert action can change display, scope, measurement, or downstream validity in different ways.[1]

Paper 8 extends those requirements into component handoffs.

A micro-engine does not simply receive “an image.” It receives a scientifically qualified input state.

Depending on the operation, that state may include the identity of the source, dimensional meaning, physical calibration, channel role, analytical scope, accepted upstream objects, processing state, validation state, user decisions, or other prerequisites already established by the governed route.[1–3]

Likewise, an output is not merely “a result.” It may be:

- a measurement with units and parameters;
- a new analytical object set;
- a refined scope;
- an accepted or rejected intermediate state;
- a validation result;
- a limitation;
- a requirement for scientist review;
- a reroute;
- a hold;
- a blocked outcome;
- or a structured handoff to the next eligible component.

The exact implementation payloads remain controlled engineering information. The governing scientific rule is simpler:

A component handoff must preserve enough state to explain why the next component is scientifically allowed to act.

That requirement prevents composition from becoming a chain of undocumented assumptions.

6. Applicability comes before composition

A system with many analytical components can become dangerous if composition is driven only by availability.

The presence of a micro-engine does not make that micro-engine applicable.

Paper 6 established the difference between technical capability and scientific eligibility.[2] Paper 7 brought that distinction into the user-facing capability surface.[1] Paper 8 carries it into orchestration.

Before a component joins an active route, the route must determine whether the source evidence and current accepted state satisfy the component's scientific prerequisites.

A time-analysis component should not join a static source merely because the code can execute. A volumetric component should not join a route when the required dimensional evidence is absent or unresolved. A physical measurement should not proceed when calibration is missing. A downstream object analysis should not inherit a mask or object state that has become stale after expert modification. A modality-specific operation should not be inferred from visual appearance alone.[1–3]

This is why BENNETT's user-facing capability model is evidence-aware.

Capabilities can be supportable now, supportable after a prerequisite, dependent on a missing scientific mapping or expert decision, not applicable to the active evidence, blocked by a mandatory condition, or unsupported in the current validated platform state. The user may express the scientific objective, but neither the user nor the language model creates applicability by asking for a route.[1,8]

The route is composed **after** scientific eligibility is established, not before.

7. Dependencies are part of scientific meaning

Composability requires dependencies, but a dependency is more than an implementation-order convenience.

A downstream measurement may depend on an accepted upstream scope. A spatial relationship may depend on object definitions produced earlier in the route. An interpretation may depend on a measurement remaining current. A validation result may be specific to the exact state on which it was performed.

If the upstream state changes, the downstream consequence must be knowable.

This is the architectural continuation of Paper 7's return-to-governance requirement. When an expert user changes a threshold, mask, ROI, Z range, object set, or other measurement-affecting state, downstream results cannot remain current merely because they

still exist in memory or on screen.[1]

Dependency-aware composition allows the system to distinguish at least three cases:

- **Unaffected work remains current.** A change that does not alter a component's prerequisites should not trigger unnecessary recomputation.
- **Dependent work becomes stale.** A result may still exist as historical state while losing authority as the current result.
- **The route must be reconsidered.** A change may invalidate eligibility itself, requiring a new prerequisite, reroute, scientist review, or block.

The important point is selective consequence.

A governed architecture should not invalidate the entire experiment every time one state changes. It should invalidate the work whose scientific basis changed and preserve the rest.

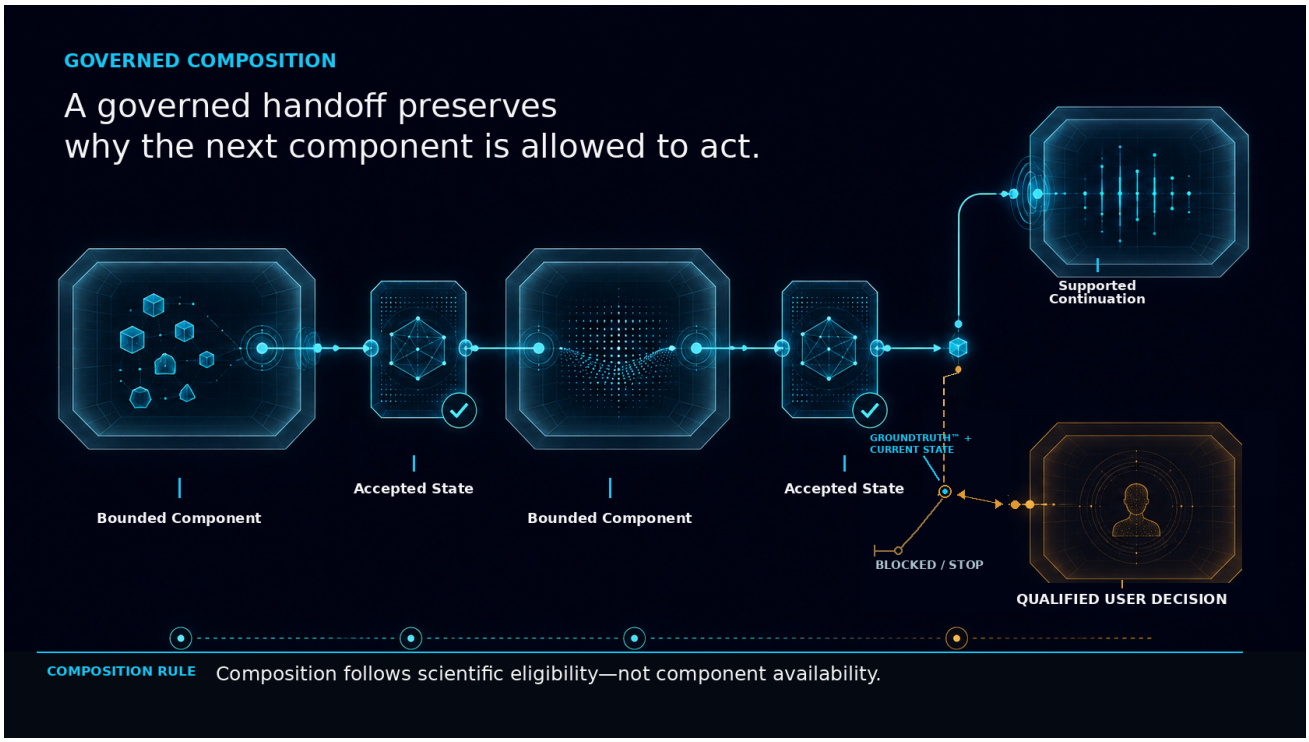


Figure 3. Composition is a governed handoff, not a blind chain. Accepted state determines whether the next bounded component is supported, requires qualified-user judgment, or must block or stop.

8. State change should propagate only where dependency exists

A scientific system becomes difficult to trust when small changes cause invisible global consequences—or when large consequential changes cause no downstream response at all.

Micro-engine architecture creates a middle path.

Because each component has bounded inputs and outputs, a state change can be evaluated against known scientific dependence. If a visualization-only action changes how evidence is displayed, quantitative components may remain current. If an analytical scope changes, components that depend on that scope may become stale. If a threshold changes object identity, object-derived measurements and interpretations may need to rerun or revalidate. If a calibration required for physical measurement becomes invalid, pixel-level source evidence may remain intact while physical measurements lose current authority.[1,2]

This is not merely computational optimization. It is scientific causality.

The route should preserve:

- what changed;
- which component or scientist action changed it;
- the prior state;
- the new state;
- which outputs depend on the changed state;
- which outputs remain independent;
- what must be rerun;
- what must be revalidated;
- and what claim boundary follows from the new state.

The exact dependency representation is implementation-controlled. The governing requirement is that dependencies must be explicit enough to prevent stale authority from passing downstream unnoticed.

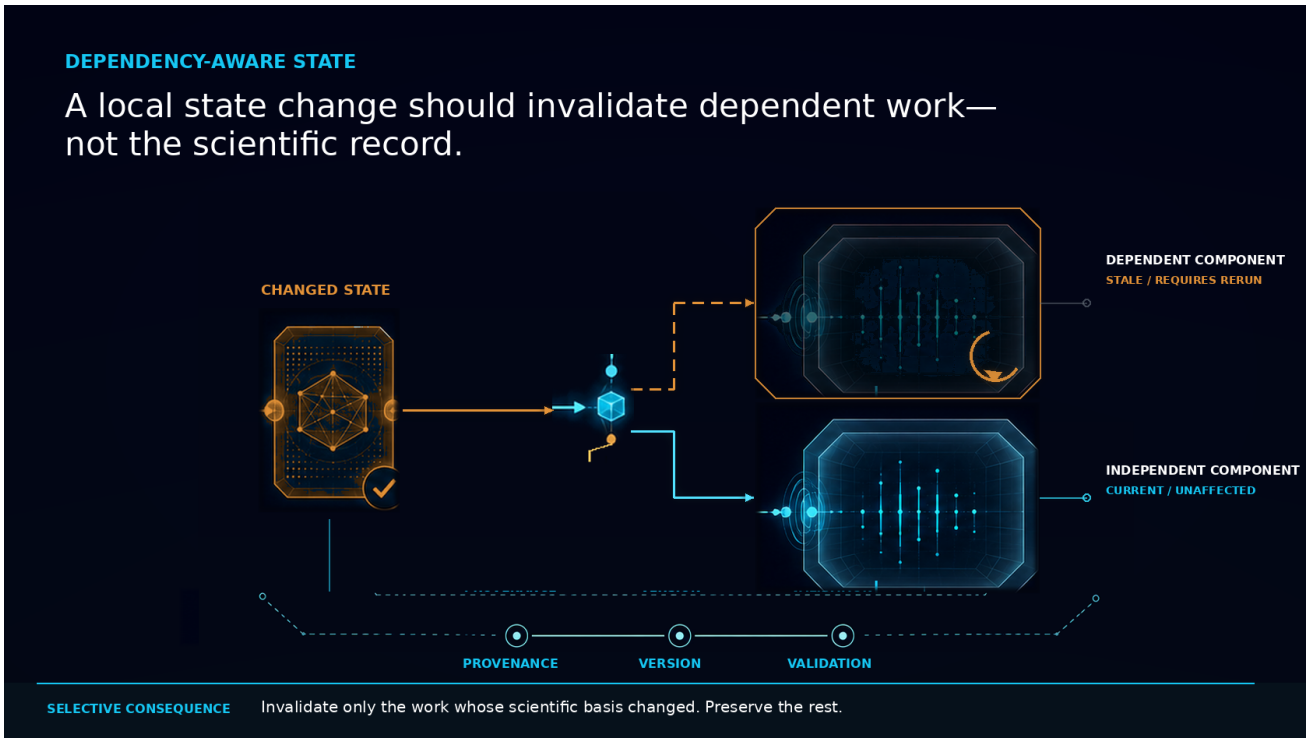


Figure 4. A local state change should invalidate dependent work, not the scientific record. Only downstream work whose scientific basis changed becomes stale; independent work and the preserved provenance, version, and validation foundation remain current.

9. Validation belongs to bounded responsibility and to the route

A monolithic scientific confidence score is attractive because it is simple. It is also capable of hiding important differences.

Paper 6 established that evidence is not validation, measurement is not validation, scientist confirmation is not validation, and a previous validation state cannot be silently reused as current authority.[2]

Micro-engine architecture preserves that distinction by making validation obligations attach to defined responsibility and scope.

A component can be tested for whether it receives the required kind of input, performs the defined operation, produces the expected output, preserves prohibited boundaries, handles negative paths correctly, and refuses scientifically unsupported states. That establishes component-level behavior within the tested scope.

A composed route creates additional questions. Even if each component functions correctly in isolation:

- were the handoffs scientifically compatible?

- were the required upstream states current?
- did a user intervention create a new state after an earlier validation?
- did the route preserve provenance?
- did a later component interpret an earlier output within its intended meaning?
- did the final claim remain within the boundaries of the complete route?

Validation therefore exists at more than one level without becoming one undifferentiated score.

The component should know its bounded obligation. The route should know whether those obligations remain satisfied together.

10. Negative paths and failure isolation are part of composability

A component architecture is not governed merely because successful components can be chained together.

It must also compose failure correctly.

Paper 6 established negative paths as legitimate scientific outcomes: insufficient evidence, wrong scope, not applicable, review required, result limited, route blocked, claim unsupported, or stop.[2] Paper 7 applied that logic to direct scientist operation.[1]

Micro-engines make those outcomes localizable.

If a component cannot operate because a required state is missing, the architecture should preserve why it cannot operate and which next actions remain scientifically legitimate. The failure should not be converted into a generic model response that chooses a plausible substitute. Nor should one local negative path automatically invalidate unrelated evidence.

This creates a practical distinction among:

- a **component failure**, where the intended operation did not execute as designed;
- a **scientific block**, where execution would be inappropriate under the active evidence;
- a **conditional route**, where a defined prerequisite or expert decision can resolve the gap;
- an **unsupported capability**, where the current governed implementation does not authorize the requested route; and
- a **valid negative result**, where the scientifically correct outcome is absence, limitation, non-evaluability, or stop.

A robust architecture must preserve those differences because each has a different scientific consequence.

11. Versioning makes scientific capability amendable without becoming fluid

Scientific methods change. Software changes. Instruments change. New evidence exposes failure modes that were not visible earlier. A governed platform must therefore be amendable.

Amendability is different from silent mutability.

A bounded component can be versioned because its responsibility is identifiable. A revised component can declare that its method, accepted inputs, output behavior, validation state, or boundary has changed without pretending that all prior results were produced under the new state.

This matters for scientific reconstruction.

A result should remain attributable to the component and method version that actually produced it. If a later version changes analytical behavior, the system can determine whether prior outputs remain comparable, require rerun, or should be interpreted under a different boundary.

The same principle applies to the route. A new component or amended component should not silently rewrite the behavior of unrelated validated components. Changes should enter through explicit, reviewable amendments rather than through an invisible global rewrite.[8]

This is one reason composability and governance reinforce one another. Smaller scientific responsibilities can be changed locally while the platform-wide evidence obligations remain stable.

12. Evidence-aware routing exposes outcomes, not the internal registry

Paper 7 established the user-facing consequence of file-aware routing: the scientist should see analytical capabilities that the active evidence can actually support rather than an undifferentiated list of everything the platform contains.[1]

Paper 8 explains what that implies underneath.

The user chooses a scientific outcome. The system resolves the bounded components and prerequisites required to perform that outcome under the active evidence state. The user should not need to know every internal component identifier or manually assemble every dependency.

A capability may be recognizable through overlapping scientific facets such as dimensionality, time structure, modality, channels, physical calibration, object state, or

localization-compatible evidence. These facets are not mutually exclusive buckets. A dataset may be simultaneously three-dimensional, time-resolved, multichannel, multi-position, and modality-specific.[8]

The scientific objective remains human-legible:

- measure a spatial relationship;
- locate and quantify objects;
- inspect a temporal change;
- review an accepted segmentation;
- perform a modality-appropriate refinement;
- or another defined scientific outcome.

The internal route may involve several bounded components. The user does not need to convert the scientific question into an engineering diagram before BENNETT can act.

The important governance rule is that outcome-oriented UX does not erase component responsibility. The user-facing interface can remain simple while the execution record remains specific.

13. Micro-engines make the LLM narrower by design

Paper 7 argued that the language model becomes more useful when its scientific authority is narrower.[1]

Micro-engine architecture makes that separation operational.

Stable scientific logic does not need to live in conversational prose. **This is not merely an efficiency choice. It is an anti-drift architecture.**

A conversational model is powerful at language and contextual reasoning, but it is a poor place to store scientific obligations that must resolve the same way every time. If eligibility rules, dependencies, negative paths, validation requirements, state consequences, and claim boundaries must be reconstructed from prose on each turn, the system is repeatedly rebuilding its own scientific operating state. That invites omission, inconsistent reconstruction, and model-dependent drift.

BENNETT removes that burden from the LLM. The governed system resolves the active evidence and applicable route first. Stable scientific responsibilities remain in bounded components whose prerequisites, operations, state consequences, negative paths, provenance, and validation obligations are explicit and testable. The conversational layer receives the scientific context required for the current objective rather than carrying the platform's governing logic as conversational memory.[1,2,8]

The LLM can:

- understand the user's stated objective;
- clarify terminology;
- ask for a missing mapping or decision;
- explain why a route is available, conditional, blocked, or unsupported;
- explain the result returned by deterministic components;
- and maintain conversational continuity around the governed state.

It does not independently determine whether a required axis exists, whether calibration is valid, whether an upstream prerequisite can be skipped, which validated analytical implementation should run, or whether a downstream claim is scientifically permitted.[1–3]

This architecture can reduce how much stable scientific logic must be carried through conversational context, but this paper does not claim a measured token-performance improvement. The more important consequence is reproducibility and testability: conversation remains flexible while scientific eligibility and execution remain bounded. The model does not have to remember or recreate the scientific architecture; it communicates with an architecture that already defines and preserves the governing state.

The language model communicates. The governed architecture preserves state, constrains, and executes.

14. Scientist-in-Control spans components without becoming another component

A micro-engine architecture must not turn the scientist into an error-handling service or a software module.

Scientist-in-Control is not simply human-in-the-loop operation, and the scientist is not a fallback invoked when automation fails. The qualified scientist is a distinct scientific authority whose legitimate judgment is represented as a first-class governed state.

Paper 7 established the opposite relationship. The qualified expert remains the scientific authority where legitimate judgment is required. BENNETT preserves the scientific conditions and consequences around that judgment.[1]

Composition must therefore support explicit scientist-decision boundaries.

A route may pause because:

- an analytical scope must be chosen;
- candidate objects require inspection;
- more than one defensible interpretation remains;
- a mapping is ambiguous;

- a supported parameter requires expert selection;
- a limitation must be accepted;
- another evidence route should be requested;
- or the scientifically correct decision is to stop.

The expert may operate directly and return a structured scientific state. The composed route then determines which downstream components remain current, which need rerun, and which validation obligations must be reconsidered.[1]

The architecture therefore separates **component responsibility** from **scientist authority**.

A component may establish what operation it can perform and what evidence it requires. It does not decide that an expert's legitimate scientific discretion should be replaced by a default merely because a deterministic implementation exists.

15. The difficult question: does decomposition just create a harder-to-manage tangle?

QUESTION

Does breaking BENNETT into engines and micro-engines simply move complexity from one large system into many smaller bounded components?

ANSWER

The complexity does not disappear. The architecture makes consequential complexity explicit, bounded, and testable.

The architecture did not begin with a desire to create many components. It reflects a recurring method in the scientific work that preceded BENNETT: distill a complex problem until each consequential responsibility can be stated, tested, and recombined without losing the logic of the whole.[2]

Micro-engines are not complexity multiplied. They are complexity distilled into governable units.

Unmanaged decomposition would still be a problem. Many components with unclear ownership, hidden dependencies, duplicated authority, inconsistent state, and ad hoc handoffs would be harder to govern than one large workflow.

The safeguard is that a BENNETT component is not considered useful merely because it is small. Its scientific responsibility must be identifiable. Its inputs and prerequisites must be bounded. Its output and state consequence must be explicit. Its negative paths must be preserved. Its provenance and version must be attributable. Its downstream dependencies and validation obligations must be reviewable. A component that cannot satisfy those conditions is not made trustworthy by being called a micro-engine.[2,8]

This architecture therefore does not eliminate complexity. It changes the form of complexity from hidden coupling to explicit responsibility. That allows a reviewer to ask which component acted, why it was eligible, what it changed, what depends on it, and what must be revalidated—questions that become harder when all responsibility is concentrated in one opaque analytical path.

16. Controlled extensibility is the central architectural advantage

The strongest reason for micro-engines is not that smaller components are fashionable or easier to draw.

It is that BENNETT must be able to grow without rebuilding its scientific foundation.

For purposes of this paper, the **Foundation Layer** is the durable set of scientific obligations already established by the collection:

- preserved evidence and source lineage;
- explicit analytical state;
- provenance of consequential operations and decisions;
- validation obligations;
- expert control where legitimate judgment is required; and
- claim boundaries that prevent output language from exceeding the route that supports it.[1–5]

A new analytical capability should inherit those obligations.

Adding a new micro-engine may add a new measurement, refinement, analysis, or route. It may introduce new prerequisites, new output types, new negative paths, or new validation requirements. What it must not do is silently redefine what counts as source evidence, erase prior state, create scientific applicability through technical availability, bypass provenance, convert scientist confirmation into validation, or broaden claim authority beyond its tested scope.

This is governed expansion.

The supportable capability surface becomes larger for evidence states that can legitimately use the new component. Evidence states that do not meet the component's requirements remain unchanged. The governing scientific foundation does not need to be authored again merely because the platform gained a new analytical function.[1,8]

The result is an architecture that can be both stable and extensible:

stable in scientific obligation; extensible in analytical capability.

17. What the current BENNETT architecture can responsibly establish

BENNETT's controlled engineering record contains parent-engine roadmaps, micro-engine specifications, use-case packages, routing logic, validation records, component UX requirements, and Publisher handoff requirements.[8]

That record supports the architectural statements made in this paper: BENNETT is designed around bounded analytical responsibilities, explicit applicability, governed orchestration of deterministic analytical components, scientist-control boundaries, stateful handoffs, negative paths, provenance, versioning, and downstream reporting obligations.

This paper does **not** claim that every planned engine or micro-engine has reached the same implementation, test, or release state.

That distinction is important. Different internal registries and engineering packages were created at different development moments, and component status must be reconciled against the current controlled source packages and validation record before any individual engine is described as built, tested, planned, or released.[8]

The architecture can therefore be explained without turning a roadmap into a performance claim.

Paper 8 establishes the architectural design logic:

- scientific responsibility is decomposed rather than concentrated;
- component boundaries are scientific boundaries;
- components receive and produce explicit scientific state;
- applicability governs composition;
- dependencies determine downstream consequence;
- local changes create selective invalidation rather than silent global drift;
- validation remains scoped;
- negative paths remain legitimate outputs;
- versions remain attributable;
- the LLM remains conversational rather than becoming the hidden scientific operating system;
- scientist control remains first-class; and
- new capability can be added without rebuilding the Foundation Layer.

The implementation details that operationalize these principles—including internal registries, dependency representations, routing logic, eligibility predicates, validation fixtures, and source code—remain controlled engineering information.[8] They are not required to evaluate

the scientific architecture described here.

18. Handoff to Paper 9 — The BENNETT Scientific Publisher

A composed analytical route creates one final responsibility that Paper 8 should not absorb.

Once multiple bounded components have operated on preserved evidence, the system must produce more than a final number or image.

A defensible downstream record may need to preserve:

- the native source and relevant experimental context;
- the analytical components and methods that acted;
- the active scope and state transitions;
- measurements and their current/stale status;
- visual review states where scientifically consequential;
- scientist actions and decisions;
- validation status;
- limitations and negative paths;
- provenance;
- and the claim boundaries attached to the route.[1–4,8]

The number of components in the route should not weaken that obligation. A route containing a newly added capability must be just as reconstructable as a route using a long-established capability.

That creates the next question:

> How does BENNETT assemble the evidence, methods, measurements, scientist actions, provenance, validation status, limitations, and claim boundaries produced across a governed route into one defensible scientific record—without allowing the report itself to become a new source of scientific authority?

Paper 9 answers that question through **The BENNETT Scientific Publisher**.

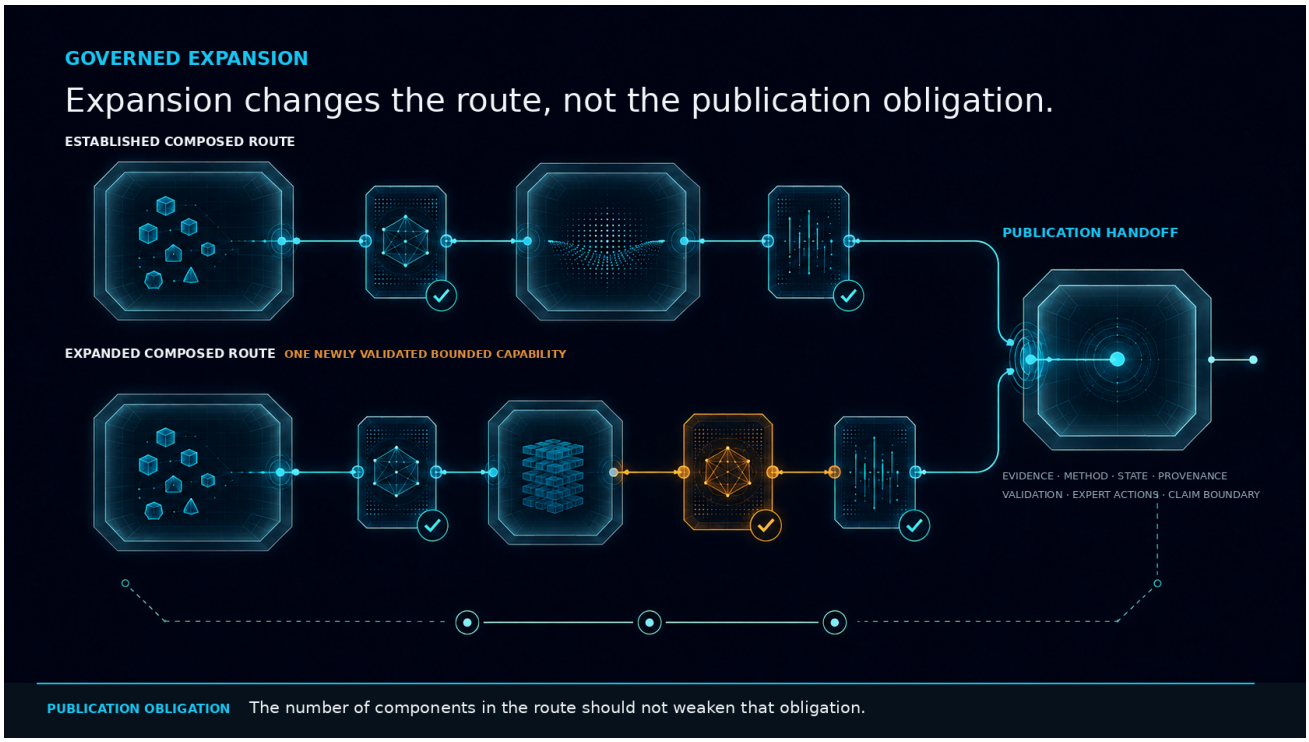


Figure 5. Expansion changes the route, not the publication obligation. An established route and a route containing one newly validated bounded capability converge on the same structured handoff obligation: evidence, method, state, provenance, validation, expert actions, and claim boundary.

Conclusion

A scientific platform becomes difficult to govern when analytical capability and scientific authority are allowed to accumulate in the same opaque place.

BENNETT separates them.

The platform can contain many analytical capabilities without treating any one model, engine, micro-engine, or workflow as the source of all scientific authority. Native evidence remains authoritative for acquisition. The complete experiment remains the context that gives the evidence meaning. Deterministic analytical components perform bounded operations. Applicability determines which components may join the route. Dependencies determine which results remain current when state changes. Validation remains scoped to defined obligations. Negative paths remain visible. Legitimate scientist judgment remains a distinct scientific authority rather than becoming a model default or fallback. The language model remains conversational and explanatory; stable scientific obligations, prerequisites, state, and downstream consequences remain in the governed architecture rather than being reconstructed from conversational context.[1–7]

Micro-engines are the implementation expression of that separation.

Their value is not that the platform contains many small pieces. Their value is that scientific

responsibility becomes addressable. A component can be tested because its job is bounded. A result can be reconstructed because its state and provenance are explicit. A change can be propagated because dependencies are known. A failure can be localized because negative paths remain meaningful. A new capability can be added because the scientific foundation does not have to be reinvented around it.

The architecture therefore solves the expansion problem inherited from Paper 7.

BENNETT can add new scientifically bounded analytical capability on top of a stable governed foundation without rebuilding or redefining that foundation each time the platform expands.

That does not make expansion automatic. Every new capability must establish its own supported scope, prerequisites, state consequences, validation obligations, negative paths, provenance, and claim limits before it can carry scientific authority in a route.

The principle is simple:

Add capability locally. Preserve scientific authority explicitly. Keep the governing foundation stable.

Paper 9 now takes the composed route and asks how its evidence becomes a defensible scientific publication object.

References

1. Bennett BT. Scientist-in-Control AI. BENNETT White Paper 07. 2026. https://486b4e8f-f0c1-4b95-8447-81ae78bc5540.filesusr.com/ugd/175885_1cc2eb114dd54f499fa2792214d6644a.pdf

2. Bennett BT. From Expert Practice to Governed Scientific Intelligence. BENNETT White Paper 06. 2026. https://486b4e8f-f0c1-4b95-8447-81ae78bc5540.filesusr.com/ugd/175885_0cf9c0fd6a0142318b3340915e08a85a.pdf

3. Bennett BT. The Native File as the Acquisition GroundTruth™. BENNETT White Paper 05. 2026. https://486b4e8f-f0c1-4b95-8447-81ae78bc5540.usrfiles.com/ugd/175885_c0be36ea34504c20bc50839fdc6738ec.pdf

4. Bennett BT. The Complete Experiment. BENNETT White Paper 02. 2026. https://486b4e8f-f0c1-4b95-8447-81ae78bc5540.usrfiles.com/ugd/175885_e10f44f16e9b4a98ba19c5596f490715.pdf

5. Bennett BT. From 4Pi Microscopy to Governed Scientific Intelligence. BENNETT White Paper 01. 2026. https://486b4e8f-f0c1-4b95-8447-81ae78bc5540.usrfiles.com/ugd/175885_e1d23cf6bee342279b8386aee3f0df94.pdf

6. Bennett BT. Resolution Stepdown. BENNETT White Paper 03. 2026. https://486b4e8f-f0c1-4b95-8447-81ae78bc5540.filesusr.com/ugd/175885_e1b203fc28f54cf5a66be6adb01c4b3f.pdf

7. Bennett BT. Antibodies You Can Trust. Data You Can Defend. BENNETT White Paper 04. 2026. https://486b4e8f-f0c1-4b95-8447-81ae78bc5540.filesusr.com/ugd/175885_0ca91b9a2ade459188a663e57fd8f8f7.pdf

8. BENNETT controlled engineering library. Parent-engine and micro-engine specifications, use-case and capability-routing architecture, state/provenance requirements, validation controls, and Publisher handoff requirements. Internal controlled records, 2026. Implementation details intentionally omitted to preserve the proprietary implementation boundary.

9. OpenAI. Models. OpenAI API documentation. 2026. https://platform.openai.com/docs/models/gpt-4-turbo-and-gpt-4

10. Anthropic. Claude Platform documentation: tool use and Claude API skill references. 2026. https://docs.anthropic.com/it/docs/about-claude/models/migrating-to-claude-4