

Establishing IN THE WHACK™ @ InTheWhack.com as The First Instantiation of the Inverted Hexagonal Clean Bubble Architecture: A Proposed Advancement Toward a More Data-Static Agentic Future for Us All

Abstract—What We Have Found So Far

Modern software architecture remains strongly organized around applications, processes, APIs, schemas, and the continuing movement and reshaping of data among them. As Artificial Intelligence becomes increasingly agentic, however, this process-centered orientation may become less useful than an architecture in which data remains substantially more static while adaptive intelligence moves in it, through it, and around it—while avoiding the bloat experienced during the OOAD, OOP and OOD phase of our recent application and systems-engineering history.

This paper proposes an Inverted Hexagonal Clean Bubble Architecture: a recursive, data-static architectural model in which canonical data is persisted through immutable, versioned representations containing ephemeral or governed contents, while dynamically instantiated WHAT and HOW bubbles provide internal transformations and outwardly accessible representations as needed.

The model retains important principles from Hexagonal Architecture and Clean Architecture—especially separation of concerns, ports, adapters, and protection of the domain core—but reorganizes their visual and functional emphasis around concentric spheres of knowledge rather than flat hexagons.

It also inverts the component relationships, placing data at the centermost layer within the bubble and instantiated process graphs around various portions of the various layers of the bubble as needed.

At the center of every Primary Bubble is its canonical **WHY**.

The nature and contract of that canonical representation are immutable within a version. Its contents may be ephemeral, changing, superseded, contested, refreshed, or replaced according to explicit governance.

And when the representation itself must change, it changes through explicit versioning rather than silent structural mutation.

Surrounding WHY is a HOW layer responsible for transformations.

The outermost WHAT layer represents the bubble to the known world.

Agentic systems discover, identify, persist, adapt, and ultimately enmesh data into a fluidly persistent and porous fabric capable of detecting, locating [according to provenance], protecting, transforming, monetizing, sharing, and reusing information while leaving its canonical meaning increasingly undisturbed as it progresses from WHY to WHY among its neighbors.

Building IN THE WHACK™: An Architectural Summary —Version 0.1

Where meanings, identities, versions, authorities, or representations disagree, the architecture does not require premature convergence.

Instead, **Divergence™** may be allowed to reign: contradictory or unresolved assertions may coexist, each retaining provenance, confidence, authority, and context until stewardship, additional evidence, or Agentic growth can overcome the diversigent anomaly.

The aim is a future in which we remove dependencies and unnecessary coding by moving computational processes in, through and around data rather than continually forcing data into the temporary forms demanded by application processes that hold data only at their edges. In this way data is the only primary citizen, because even processes are themselves data instantiations—instantiations that, when unpacked in the context of their surrounding neighborhood graph, produce the same effects that current process-centric models strive for today at less than optimal costs.

By observing an orthogonal data posture, the Inverted Hexagonal Clean Bubble Architecture may help future generations of convergent Agentic engineers achieve increasing levels of kaizen that are difficult to attain in a world where rules and processes are the primary architectural citizens at the expense of the data that instantiates our world. For we are nothing without data.

And we would all be better off if data moved only as much as it needed to in the end—remaining at rest, or at least at maximum useful inertia, whenever possible.

1. From Application-Centric Data to Data-Static Agentics

Traditional software systems generally begin with an application.

The application establishes schemas, owns representations, defines interfaces, transforms incoming information, stores it in application-specific forms, and later exports or transforms it again for other systems.

Over decades this has produced an enormous universe of duplicated, transformed, migrated, serialized, deserialized, synchronized, cached, and repeatedly reinterpreted data. Agentic Artificial Intelligence suggests another possibility.

Instead of asking:

How do we continually move and reshape data so that applications can use it?

We may increasingly ask:

How can data remain canonical and substantially static while intelligent capability moves toward it and temporarily adapts around it?

This paper proposes an architecture intended to explore that possibility.

The central idea is simple:

Keep WHY as canonical and static in representation as possible. Instantiate HOW and WHAT only as needed.

Static does not necessarily mean unchanging contents.

A WHY representation is immutable within its version, while the values represented within it may remain ephemeral, temporal, governed, or explicitly superseded.

When the representation itself requires change, it does not quietly mutate.

It versions.

Agentics then becomes less about owning data and more about discovering, interpreting, locating, adapting, connecting, and stewarding it—even when that data is a process at its heart.

2. Why Bubbles Rather Than Hexagons

Hexagonal Architecture provides an extremely useful conceptual separation between a system's internal domain and the adapters through which external systems interact with it.

But hexagons are fundamentally flat abstractions.

They are excellent at communicating:

- boundaries,
- ports,
- adjacency,
- adapters,
- inward and outward dependency.

They are less expressive when the architecture itself depends upon:

- depth,
- nested knowledge,
- recursive composition,
- membranes,
- interior and exterior representation,
- variable proximity,
- spheres within spheres.

For this reason, the proposed primitive is not a hexagon but a bubble.

A bubble is a sphere of knowledge.

It has an outside.

It has an inside.

It can contain other bubbles.

It can exist beside other bubbles.

It can appear at one scale as a single thing and, when examined more deeply, reveal an entire collection of canonical things within it.

The Hexagonal architectural principles remain.

The nature of the components is inverted and the geometry is deepened.

3. The Three-Layer Bubble Primitive

Every Primary Bubble follows the same three-layer anatomy:

WHY — The Canonical Core

At the center is WHY.

WHY contains the canonical representation of the thing itself.

It may be:

- one canonical datum,
- a record,
- a document,
- a conversation,
- a person,
- a machine,
- a dataset,
- a graph,
- a process,
- or a collection of other Primary Bubbles.

The important property is not simplicity.

The important property is canonicity.

A complex WHY may therefore be a canonical bubble containing many canonical bubbles.

The nature of the canonical representation is immutable within a version.

The contents represented by it may be ephemeral.

If the nature of the representation changes, a new version is instantiated.

Thus canonicity does not require pretending that reality stands still.

It requires that structural meaning not be silently rewritten beneath the data that depends upon it.

The center answers:

What is this thing independent of the temporary form in which another system wishes to encounter it?

WHAT — The Known-World Layer

The outer layer is the representation of the bubble to the known world.

It contains forms appropriate for neighboring systems, devices, humans, bots, protocols, applications, interfaces, and other recognizable external environments.

Building IN THE WHACK™: An Architectural Summary —Version 0.1

Examples might include representations shaped for:

- email,
- messaging,
- relational systems,
- APIs,
- application HMLs,
- documents,
- legacy systems,
- bots,
- human-facing interfaces.

The WHAT layer answers:

In what form must I appear so that this neighbor can encounter me?

And:

Where is it that I may be found?

HOW — The Transformation Layer

The middle layer contains the transformations needed to move between canonical meaning and outward representation.

HOW owns neither canonical meaning nor knowledge of real world existence.

HOW's purpose is maintaining equivalence.

More precisely, its purpose is the most formally equivalent result attainable without functional divergence, produced in the least functionally distorting and lowest-cost manner appropriate to the need.

This Aristotelian aspiration is not superficial sameness in a final way.

It is preservation of the **WHAT** the thing is in the appropriate formal representation **for the purpose at hand**, with the least violence to its final nature.

HOW answers:

How can my canonical WHY be represented faithfully in the form required by a particular WHAT?

HOW bubbles may be resident, remotely available, newly synthesized, or temporarily injected.

They will most decidedly be combinations of the kinds of procedural code and assertive transformational statements we know and use today.

Transformations should therefore be capable of declaring properties such as:

- exact or approximate,
- lossless or lossy,
- reversible or irreversible,
- deterministic or probabilistic,
- authoritative or derivative,

Building IN THE WHACK™: An Architectural Summary —Version 0.1

- semantically equivalent or purpose-equivalent,
- expected cost,
- expected distortion,
- confidence.

A transformation that is merely useful must not silently represent itself as exact.

This gives the architecture its simplest expression:

WHY is canonical.

HOW establishes equivalence and transcendental need.

WHAT establishes exterior compatibility and discoverability.

4. Primary and Secondary Bubbles

The architecture distinguishes between two fundamental bubble classes.

Primary Bubbles

A Primary Bubble possesses independent canonical meaning.
Its WHY is intrinsic.

A Primary Bubble may contain other Primary Bubbles within its canonical WHY.

Primary Bubbles may also exist in fabric proximity with other Primary Bubbles and form meaningful canonical relationships.

A Primary Bubble represents **being**.

Secondary Bubbles

Secondary Bubbles exist because a transformation, representation, or architectural mechanism caused them to be instantiated.

Their WHY is therefore much simpler.

Their reason for existence is essentially:

Represent this Primary Bubble in this particular form under this particular transformation.

A Secondary Bubble does not independently acquire a rich recursive WHY merely because it is useful.

If a Secondary representation later becomes independently meaningful, it may be treated as a new Primary and flattened into the same three-layer primitive.

This prevents uncontrolled recursion through transformations.

The distinction can therefore be summarized as:

Primary Bubbles have an intrinsic WHY.

Secondary Bubbles have a derived and flattened WHY.

Or:

Primary = being.

Secondary = representing.

Transformation = becoming.

5. Canonical Ports

A pure Primary Bubble association has only three canonical passage points corresponding to three layers on either side of the paired bubble relationship.

These may be understood as special transparent Secondary Bubbles.

They do not alter canonical meaning.

They provide controlled passage through the architecture.

The three canonical passage points are:

1. WHAT canonical port
2. HOW canonical port
3. WHY canonical representation

The WHAT and HOW canonical ports are transparent.

They may:

- inspect metadata,
- unpack membrane instructions into helper bubbles,
- determine whether a transform is required,
- locate an existing helper,
- request a helper,
- instantiate a helper,
- enforce rights,
- evaluate freshness,
- evaluate version,
- establish routing,
- evaluate confidence,
- detect divergence.

But they do not themselves semantically transform the canonical payload.

The WHY remains pure canonical form.

This yields a useful rule:

Three canonical ports. Infinite instantiated expressions.

The Primary anatomy remains fixed even as its outward capabilities vary indefinitely.

Only neighbors with contextually congruous codices and versions can share their canonical essences transparently in this way.

Contextually incongruent neighbors must find different external ports through which to share and interact.

6. The DNA Message

All communication through the architecture is expressed as a common message primitive called DNA.

At its simplest:

DNA = Metadata + Payload

Metadata may include information such as:

- identity,
- Object GUID,
- Instance GUID,
- codex,
- codex version,
- representation version,
- source,
- provenance,
- timestamps,
- lineage,
- asserting authority,
- authority scope,
- confidence,
- evidence,
- rights,
- permissions,
- routing,
- freshness expectations,
- transform requirements,
- transformation properties,
- monetization assertions,
- state,
- connection preferences,
- revocation information,
- policy references.

The payload carries the underlying content.

The architecture does not require every payload to be physically identical.

Instead, the DNA contract provides a stable envelope within which increasingly diverse canonical information can travel.

DNA therefore carries not simply data, but enough context for the fabric to know what is being asserted, by whom, under what authority, according to which representation, with what confidence, and under which conditions it may be used.

7. Codices, Versions, and Pure Primary Communication

Two Primary Bubbles may communicate through the pure canonical path only when they recognize one another as belonging to the same sufficiently congruous DNA codex and applicable version.

This is stricter than superficial compatibility.

If two WHAT canonical ports match exactly under the same codex and relevant version, the canonical path may remain transparent.

The route still follows every layer:
WHAT → HOW → WHY

But no semantic transformation is necessary.

The WHAT and HOW canonical ports simply pass the canonical DNA inward according to their membrane rules.

This establishes an important invariant:

There is no level skipping.

Even in the perfect codex match, a message does not jump directly from outer contact to WHY.

It follows the same path as every other encounter.

Versioning applies at every level where representational change may matter.

A version mismatch is itself meaningful DNA.

It may trigger:
compare → evaluate → refresh → reconcile

If suitable refresh is unavailable, the relationship need not collapse immediately.

Its confidence or fitness may instead **depreciate diversigently** until a compatible version, transform, helper, or stewardship decision becomes available.

Thus a version mismatch is not necessarily an error.

It is a condition requiring interpretation.

Codices themselves may be persisted as Primary Bubbles.

A codex can therefore possess:

- identity,
- provenance,
- ownership,
- versions,
- compatibility relationships,
- forks,
- deprecations,

Building IN THE WHACK™: An Architectural Summary —Version 0.1

- lineage,
- rights,
- HOW mappings to other codices.

The language through which bubbles understand one another is therefore itself capable of being understood by the architecture.

8. Adapted Communication Between Different Codices

When two Primary Bubbles differ meaningfully in codex or applicable version, the pure canonical route cannot be assumed.

By definition, their WHAT ports do not match sufficiently.

One or both Primaries must therefore provide adapted WHAT and, when necessary, HOW representations.

Normally these Secondary representations will already be available for regularly encountered neighbors.

For an unfamiliar neighbor, the required WHAT or HOW may need to be acquired from elsewhere in the fabric.

The sequence remains:

WHAT → HOW, if needed → WHY, if needed

Never:

WHAT → WHY

or:

external system → HOW

or:

external system → WHY

A message may terminate at WHAT.

It may proceed through HOW.

It may eventually reach WHY.

But no membrane may be skipped.

This gives every architectural layer the opportunity to apply its own constraints.

9. Resident, Hard-Kept, and Soft-Kept Capability

A bubble should not need to carry every adapter it might ever use.

That would make bubbles unnecessarily heavy.

The architecture therefore permits WHY, HOW and WHAT capability to exist under different residency patterns.

Resident or Hard-Kept

A commonly needed adapter may be retained close to the Primary Bubble.

This may be appropriate when:

- use is frequent,
- latency matters,
- the source is unreliable,
- reacquisition is expensive,
- the neighbor is routinely present,
- licensing encourages local retention.

Soft-Kept

A rarely used helper may remain remotely available.

The bubble retains knowledge of how to find it rather than retaining the executable representation itself.

When needed:

locate → acquire → validate → instantiate → use → release

Remote Soft Injection

A helper may also remain instantiated elsewhere and be softly connected into the appropriate membrane location for temporary use.

In this case the bubble obtains capability without permanently holding the helper.

This allows the Primary Bubble to remain lightweight while still possessing access to potentially enormous amounts of distributed capability.

Handling the WHY

This same pattern also applies to the central WHY inasmuch as core meaning data can be either hard-kept—and appropriately refreshed—for practicality's sake—or dynamically injected through a pure transport when needed.

Data represented at its authoritative source of creation may be hard-kept without a conventional refresh requirement because that source is already the most resident known expression of the datum.

Remote applications that wish to share this ultimately resident data may maintain hard or soft connections and their corresponding fixed/refreshed or dynamically injected storage patterns.

Building IN THE WHACK™: An Architectural Summary —Version 0.1

This does not mean the bytes can never move.

The aim is instead:

Data at rest—or at least at maximum useful inertia.

Move canonical matter only as much as function, economics, resilience, law, or physics require.

Move capability toward it whenever doing so is superior.

10. Predictive Agentics and Residency

Agentic systems and subsystem components thereof may determine when HOW and WHAT helpers should remain hard-kept and when they should remain soft-kept.

The decision can depend upon factors such as:

- encounter frequency,
- predicted recurrence,
- reacquisition latency,
- storage cost,
- licensing terms,
- privacy requirements,
- reliability,
- expected neighbor behavior,
- freshness demands,
- version compatibility,
- previous failures.

A useful operating principle is:

Keep close what the bubble is likely to need. Keep reachable what it probably will not.

Residency therefore becomes an adaptive Agentic decision without changing the Primary Bubble's fixed anatomy.

11. Reuse Before Creation

Before producing a new adapter, Agentics should always ask:

Do we already have one of these?

The preferred order is:

Reuse → Adapt → Create

The fabric may search:

- resident helpers,
- soft-kept helper locations,
- nearby fabric persistence,
- broader fabric catalogues,
- previously instantiated transformations,
- similar codex mappings,
- version-compatible relatives,
- diversigently useful but imperfect helpers.

Only when no suitable existing helper can be found should a new one be synthesized.

A helper need not be discarded merely because it is imperfectly matched.

It may remain discoverable with a lower confidence or fitness factor.

Its usefulness may depreciate diversigently rather than disappear absolutely.

This enables architectural knowledge to compound.

A connection solved once becomes a candidate solution for every later encounter.

12. The Factory

When no suitable WHAT or HOW helper exists, the architecture may consult a factory.

A factory is a location capable of supplying or constructing codex-aware transformations and representations.

A factory may contain:

- known helper templates,
- reusable mapping patterns,
- executable components,
- source representations,
- tests,
- validation instructions,
- provenance,
- rights metadata,
- monetization terms,
- self-organizing foundry agents,
- steward-in-the-loop assistance dashboards,
- dynamic process creation and modification capabilities.

Agentic systems—including large language models—may assist factories by interpreting unfamiliar codices and proposing new adapters.

However, the model does not permit Agentics to redesign the Primary Bubble anatomy arbitrarily.

The Agentic system may create something that fits a known membrane location. It may not simply invent a different organism.

Thus:

The organism remains static while its repertoire grows.

13. Code as Canonical Data

Helper code itself may be persisted as canonical data.

Instead of requiring every helper to remain permanently executable, a helper bubble may contain canonical information sufficient to reassemble executable behavior when needed.

This may include:

- source,
- intermediate representation,
- declarative transformation rules,
- dependencies,
- build instructions,
- tests,
- expected input/output codices,
- versions,
- provenance,
- rights,
- author identity.

In a soft-connected situation:

locate helper DNA → acquire → validate → assemble → instantiate → use → release

The executable instance may disappear.

The canonical helper definition remains.

This suggests a broader principle:

Code is data whose WHAT can become behavior.

14. Simple, Complicated, and Complex WHY Instantiation

The physical storage mechanism underlying WHY need not be identical for every Primary Bubble.

Agentics may select the lightest persistence model capable of preserving canonical meaning faithfully.

A possible progression is:

Simple WHY

A conventional relational representation may be sufficient.

Complicated WHY

A richer PostgreSQL-based representation may be appropriate.

Complex WHY

PostgreSQL may be combined with vector, graph, semantic, event, temporal, or other complementary representations.

The important point is:

Complexity changes the contents and supporting substrate of WHY. It does not change the anatomy of the bubble.

Every instance still appears architecturally as:
WHAT / HOW / WHY

15. Refresh, Version Reconciliation, and Membrane Assertion

Resident capability can become stale.

Neighboring bubbles therefore need a way to express freshness and version preferences without gaining unilateral authority over one another.

A useful membrane grammar is:

SUGGEST REFRESH [[+ ASSERT] [+ ABANDON]]

The semantics are:

SUGGEST REFRESH

A fresher or more version-compatible representation is preferred.

+ ASSERT

Freshness or version compatibility is asserted as a condition of this relationship.

+ ABANDON

If the requested condition cannot be satisfied, abandon the attempted connection.

This distinction matters.

ASSERT does not mean:

“I can force you to refresh.”

It means:

“I will participate in this relationship only if this condition is met.”

The receiving bubble remains sovereign.

It may:

- accept,
- refresh,
- reconcile,
- proceed under a depreciation/confidence factor,
- request stewardship,
- decline,
- abandon.

A mismatch that cannot yet be reconciled may remain present under Divergence™ rather than being silently coerced into false compatibility.

This establishes a broader rule:

A bubble may establish conditions for its own participation. It may not secretly take control of another bubble by forced injection.

16. The Fabric

Bubbles may never need to touch directly.

Instead, they may come into immediate fabric proximity.

The fabric always remains between them.

Conceptually:

Bubble A → fabric proximity → Bubble B

This allows persistence, policy, provenance, monetization, discovery, trust evidence, and relationship history to be ubiquitous.

The fabric is therefore not merely a network transporting messages.

It is the medium in which bubble relationships become knowable.

Yet the fabric need not be one globally owned thing.

Fabric is better understood as an architectural protocol, persistence posture, and relationship medium.

There may be:

- personal fabrics,
- household fabrics,
- organizational fabrics,
- enterprise fabrics,
- device fabrics,
- regional fabrics,
- commercial fabrics,
- governmental fabrics,
- public fabrics,
- private fabrics.

Fabrics may themselves interact through governed proximity.

At an exterior scale, an entire fabric may even present itself to another fabric as a Primary Bubble.

This prevents the architecture from solving application silos by merely creating one universal silo.

The architecture may therefore federate recursively.

17. Porous to the Foreign, Impermeable to the Known

A particularly useful property of the fabric is asymmetric permeability.

Foreign Bubbles

A previously unknown bubble may move through a porous fabric.

The fabric can observe it sufficiently to begin learning:

- that it exists,
- where it exists,
- how it appears,
- perhaps which codex it resembles.

Foreign bubbles appear smooth.

They are not yet surrounded by accumulated fabric knowledge.

Known Bubbles

Once a bubble becomes known, the fabric forms around it.

It becomes fuzzy.

The fuzz represents the accumulated fabric context associated with the bubble:

- identity,
- location,
- relationships,
- adapters,
- rights,
- provenance,
- freshness,
- versions,
- history,
- monetization,
- trust evidence,
- confidence,
- wake/sleep state.

Known does not mean trusted.

Enmeshed does not mean owned.

Once enmeshed, the known bubble cannot simply bypass the fabric relationships surrounding it.

Thus:

The fabric is porous to discovery and impermeable to bypass.

And:

The fabric knows about the Bubble.

The Bubble remains itself.

18. Discover → Identify → Persist → Adapt → Enmesh

The lifecycle of a newly encountered bubble can be expressed succinctly:

Discover → Identify → Persist → Adapt → Enmesh

Discover

A foreign bubble is encountered.

Identify

The fabric determines as much as possible about its identity, codex, provenance, version, authority, and nature.

Persist

Its canonical primitive—and the fabric's knowledge of that primitive—is retained safely when policy permits.

Adapt

Appropriate HOW and WHAT representations are located, reused, or created.

Enmesh

The bubble becomes surrounded by accumulated fabric knowledge, relationships, rights, adapters, trust evidence, versions, and history.

Enmeshment does not mean ownership.

The fabric does not need to alter the bubble itself.

It alters how richly the bubble is understood and related.

Several distinctions therefore remain essential:

Discover ≠ trust.

Identify ≠ authenticate.

Adapt ≠ authorize.

Enmesh ≠ own.

Building IN THE WHACK™: An Architectural Summary –Version 0.1

19. Discovery Without Registration

The model deliberately separates discovery from registration.

A bubble does not necessarily need to be submitted to a centralized authority before it can be known.

The fabric may discover:

- devices,
- files,
- applications,
- APIs,
- message stores,
- documents,
- services,
- helper libraries,
- remote systems.

After discovery, owners or authorized actors may assert restrictions, permissions, and desires to share in the monetization of the data they choose to share with third parties.

The sequence therefore becomes:

Discover first. Determine and assert lawful use conditions afterward.

These assertions may include:

- private only,
- may be retained,
- may be shared,
- may be copied,
- soft connection only,
- hard connection allowed,
- derivative transforms allowed,
- monetized use,
- restricted use,
- expiration,
- attribution,
- privacy requirements.

A foundational distinction follows:

Discovery does not imply permission.

Building IN THE WHACK™: An Architectural Summary —Version 0.1

20. Location, GUIDs, Identity Reconciliation, and the Fabric Catalogue

To support a world of moving, replicated, sleeping, and rediscovered data, identity must be separated from location.

At minimum, the architecture may distinguish:

Object GUID

The enduring identity of the canonical thing.

Instance GUID

The identity of a particular manifestation or copy.

Extended Location URI

Where and how that instance can currently be reached.

One canonical object may therefore have multiple instances at multiple locations without losing identity.

The persistence layer can know:

- which instances belong to one object,
- where each currently exists,
- which are current,
- which are stale,
- which are sleeping,
- which are ROM,
- which may refresh,
- which are synchronized,
- which have disappeared.

Identity determination itself may remain diversigent.

Two objects independently discovered may later prove to represent the same thing.

One object may later prove to contain two things that were improperly collapsed.

The fabric should therefore preserve identity reconciliation as graph history rather than rewriting the past.

Relationships may include:

- SAME-AS,
- PROBABLY-SAME-AS,
- DISTINCT-FROM,
- SUPERSEDES,
- SPLIT-FROM,
- MERGED-INTO,
- DERIVED-FROM,
- DISPUTED-AS.

Confidence may accompany those relationships.

Building IN THE WHACK™: An Architectural Summary —Version 0.1

A merge therefore need not destroy either prior identity.

A split need not pretend the prior conflation never occurred.

The result is something more than DNS.

It is a combined:

identity registry + locator system + provenance graph + relationship memory + synchronization catalogue

Without requiring advance registration by every source—and perhaps, in an ultimate implementation, not requiring centralized registration at all.

21. Sleep, Wake, Death, Erasure, and Tombstones

Distributed things frequently disappear without ceasing to exist.

A mobile device goes offline.

An application is unloaded.

A helper instance is released.

A file store becomes temporarily unreachable.

For this reason, the default assumption should be sleep, not death.
Possible states include:

Awake

Currently reachable or instantiated.

Sleeping

Known, but presently unavailable or uninstantiated.

Dormant

Intentionally or persistently inactive.

Dead

Authoritative evidence indicates that the instance has been destroyed or permanently retired.

This is very much how a death certificate works in purely human interactions as well.

The governing principle is:

Assume sleep until there is sufficient evidence of death.

Nevertheless, the fabric may preserve the historical identity of a dead instance even when that instance can no longer be resolved, much as governmental records departments do in human terms.

But death and erasure are not the same thing.

Erased

Canonical matter has been removed because an authorized owner, governing policy, legal obligation, or other legitimate authority requires deletion.

Tombstoned

The matter itself has been erased, while policy permits the fabric to preserve evidence that an object or event once existed.

Such memory may retain identifying signatures, pseudonymous signatures, anonymized signatures, or no identifying signature at all, according to applicable policy assertions.

Thus the architecture may sometimes:

Keep the map. Destroy the matter.

At other times:

Keep only the memory that an erasure occurred.

And where policy requires:

Erase identifying memory as well.

Erasure itself should generate provenance to the extent lawful policy permits.

The architecture must be capable of remembering that stewardship occurred without defeating the very erasure it was required to perform.

22. Persistence Holds Both the Map and the Matter

The persistence layer does not merely store metadata about the fabric.

The canonical primitives themselves are persisted.

Therefore, the persistence layer contains both:

The map

AND

The matter

It may persist:

- canonical Primary Bubbles,
- canonical relationships,
- codices,
- helper definitions,
- adapter templates,
- identities,
- locations,
- rights,
- provenance,
- monetization,
- encounter history,
- instantiation history,
- graph relationships,
- trust evidence,
- version history,
- sleep/wake state,
- erasure and tombstone events.

The primitives must be simultaneously:

Safe

Durable, integrity protected, rights controlled, recoverable, and provenance preserving.

In Use

Queryable, composable, addressable, connectable, transformable, and able to participate in larger WHY structures.

Persisted does not mean archived.

Canonical primitives are kept safe so that they can remain continuously useful.

23. Conversational Fabric Stewardship

The fabric must eventually become conversant with humans and bots alike.

Consider an email stream newly discovered by the fabric.

Rather than silently publishing it, the fabric may ask:

“I have discovered this message stream. Would you like it privately retained under standard privacy restrictions, or do you wish to share it?”

The same pattern can apply to:

- email,
- SMS,
- WhatsApp,
- documents,
- messaging platforms,
- application histories,
- media collections,
- other personal or enterprise information.

The crucial principle is:

Discovered does not mean shared.

Retained does not mean exposed.

Normalized does not mean published.

Connected does not mean copied.

A human may converse naturally with the fabric.

Bots may converse through machine-readable membrane contracts.

Underneath both, the interaction remains DNA passing through governed ports.

24. Late-Bound Application Representation

Once information has been made externally orthogonal to its originating application, the decision about final application-specific representation can be postponed.

An email need not remain fundamentally “a Gmail object.”

A WhatsApp message need not remain fundamentally “a WhatsApp object.”

A text message need not remain fundamentally “an SMS object.”

A video message need not remain fundamentally “a YouTube object.”

They may all be retained canonically simply as messages in related message bubbles containing:

- sender,
- recipient,
- time,
- content,
- attachment relationships,
- provenance,
- source,
- thread relationships,
- rights,
- other metadata.

The original source representation can remain preserved as provenance.

A later application can request its preferred WHAT.

The fabric then finds or creates the necessary HOW.

Thus:

Source representation → canonical persistence → late-bound application representation

The destination application no longer needs to determine the long-term ontology of the data.

Building IN THE WHACK™: An Architectural Summary —Version 0.1

25. Rights, Authority, Monetized Instantiation, and Public Obligations

An authored HOW or WHAT helper may carry rights and economic assertions in its DNA.

The data owner or duly authorized steward is the primary rights asserter for privately controlled data.

An author may choose to permit:

- free use,
- attribution-required use,
- paid instantiation,
- per-execution payment,
- restricted domains,
- restricted organizations,
- private use only,
- derivative adaptation,
- no derivative adaptation,
- temporary invocation only,
- caching,
- hard residency.

An instantiation can therefore become a metered rights event.

The fabric may know that a helper exists without possessing or exposing the underlying artifact.

When another bubble wishes to instantiate that capability, the fabric evaluates the applicable assertions.

Rights assertions themselves should include sufficient DNA to establish matters such as:

- issuer,
- asserting authority,
- scope,
- effective time,
- expiration,
- delegation,
- revocation,
- governing policy,
- governing jurisdiction,
- public-interest exception when lawfully applicable.

Private rights are not always absolute.

Governments, courts, regulators, emergency authorities, contractual structures, and other legitimate institutions may in defined circumstances possess authority that affects how private assertions are applied.

The architecture should not encode one universal political or legal answer to such conflicts. Instead, applicable law, precedent, regulation, policy, and authority structures may themselves be represented as versioned canonical bubbles.

An Agentic legal or policy interpreter may then instantiate the most relevant current representation for the applicable jurisdiction and circumstances.

Building IN THE WHACK™: An Architectural Summary —Version 0.1

New statutes, rulings, regulations, or controlling decisions become new versioned Agentic variations upon later instantiation.

However:

An LLM does not acquire sovereign legal authority merely because it can interpret the law.

Agents may reason, identify precedent, surface conflicts, calculate confidence, and recommend action.

Contested overrides must still resolve through the lawful authority and human or institutional stewardship required by the governing jurisdiction.

This creates the possibility of a royalty-bearing capability fabric while preserving a broader principle:

Rights govern whether and how a connection may occur. They do not alter the primitive geometry of the system.

26. Trust, Threats, and Bubble Sovereignty

No fabric capable of discovering, adapting, and instantiating distributed capability can safely assume that everything it encounters is benevolent.

The architecture therefore requires an explicit trust posture.

Possible threats include:

- forged Bubble identities,
- malicious DNA,
- poisoned HOW helpers,
- fabricated provenance,
- privilege escalation,
- replay attacks,
- catalogue poisoning,
- compromised factories,
- malicious roving bots,
- impersonated codices,
- false version assertions,
- rights manipulation,
- attempts by one agent to persuade another to relax membrane protections.

The existence of a known Bubble is not proof of its trustworthiness.

The existence of provenance is not proof that the provenance is true.

The existence of a helper is not proof that the helper should execute.

The fuzz surrounding an enmeshed Bubble may therefore contain accumulated trust evidence such as:

- successful prior interactions,
- verified identity,
- signing history,
- observed behavior,
- authority attestations,
- failures,
- disputes,
- security findings,
- provenance confidence,
- helper validation results.

Trust is therefore contextual and evidentiary.

A foundational rule follows:

Known does not mean trusted.

And another:

Proximity does not grant privilege.

27. Agentic Authority and Stewardship

Agentics gives the architecture adaptive power.

That power requires boundaries.

An LLM may recognize that two codices appear almost equivalent.

It may propose a transform.

It may discover that a helper family could be generalized.

It may notice a rights conflict.

It may infer that two GUIDs probably refer to the same person.

But an inference must not silently become authority merely because an intelligent system produced it.

The governing principle is:

Agents may propose.

Policies authorize.

Deterministic mechanisms validate.

Stewards adjudicate where required.

An agent may therefore:

- discover,
- classify,
- infer,
- recommend,
- synthesize,
- test,
- compare,
- assign confidence,
- request stewardship.

It may not exceed the rights, authority, membrane policies, or governance applicable to the encounter.

This distinction allows Agentics to grow extraordinarily powerful without requiring the architecture to pretend that intelligence and authority are the same thing.

28. Divergence™ as an Architectural State

Traditional systems frequently treat disagreement as something to eliminate quickly.

One value wins.

One row overwrites another.

One schema becomes authoritative.

One merge is accepted.

One answer is returned.

Reality is often less cooperative.

Building IN THE WHACK™: An Architectural Summary —Version 0.1

The Inverted Hexagonal Clean Bubble Architecture therefore permits **Divergence™**.

Divergence is the governed coexistence of unresolved or partially resolved differences without forcing premature convergence.

Examples may include:

- two conflicting canonical assertions,
- uncertain identity matches,
- incompatible versions,
- competing authorities,
- imperfect transforms,
- stale replicas,
- contradictory provenance,
- divergent legal interpretations,
- incompatible codices,
- simultaneous updates.

Each diversigent branch may carry:

- source,
- authority,
- timestamp,
- provenance,
- version,
- confidence,
- applicable rights,
- relationship to competing assertions,
- stewardship status.

The fabric may then say:
These things disagree.

Rather than falsely declaring:
One of these things is reality.

Agentics may continue gathering evidence.

Confidence may rise or fall.

A version may refresh.

A steward may adjudicate.

A new codex mapping may reconcile the discrepancy.

A legal ruling may resolve the authority conflict.

Until then, the anomaly may remain.

This yields an important principle:

Convergence is an achievement, not an architectural presumption.

29. Concurrency and Conflict Without Forced Convergence

Two agents may encounter, update, or derive related canonical state at the same time.

A network may partition.

A hard-kept instance may become stale.

Two authoritative sources may issue contradictory successors.

A transformation may complete on one node and fail on another.

The Bubble Architecture does not pretend these classic distributed-systems problems disappear.

Instead, it separates canonical identity and representation from the substrate strategies used to reconcile them.

Consistency, transactions, synchronization, consensus, conflict-free replication, event ordering, and other mechanisms may be selected according to domain need.

When no immediate reconciliation is possible, Divergence may preserve competing branches with confidence and provenance rather than forcing a destructive winner.

A stewardship request can then become an ordinary fabric event.

Conceptually:

detect conflict → preserve branches → score confidence → seek evidence → attempt reconciliation → request stewardship if needed

The architecture therefore does not require reality to become falsely singular merely because computation prefers it that way.

30. Roving Bots

Roving bots may become some of the most powerful participants in this environment.

Because the fabric persistence layer accumulates identities, locations, adapters, failures, successes, codex mappings, sleep patterns, provenance, versions, divergence, and trust evidence, a bot arriving in a new neighborhood does not need to begin from zero.

It may ask:

- What bubbles are known here?
- What codices occur here?
- Which adapters worked previously?
- Which versions are current?
- What is stale?
- What is sleeping?
- What is reusable?
- What rights apply?
- What is diversigent?
- What requires stewardship?
- What patterns have other agents already learned?

Bots may specialize.

Some may:

- discover unknown codices,
- locate sleeping endpoints,
- update stale location knowledge,
- reconcile graphs,
- locate reusable helpers,
- optimize residency,
- identify adapter duplication,
- detect divergence,
- seek version reconciliation,
- discover emerging interoperability patterns.

The fabric therefore becomes a shared cumulative memory rather than a collection of isolated intelligent agents.

31. What an LLM Can Learn From Fabric Persistence

A large language model encountering the persistence layer may infer far more than exact database records.

It can reason over accumulated behavioral evidence.

For example:

- which codices frequently interact,
- which helpers repeatedly succeed,
- which locations are transient,
- which adapters are functionally similar,
- which transforms fail,
- which data types are emerging,
- which helper families should be generalized,
- which soft-kept capabilities should become hard-kept,
- which apparent schemas may actually represent the same concept,
- which version mismatches recur,
- which diversigent assertions are approaching reconciliation.

A deterministic system may observe:

“No exact adapter exists.”

An LLM may observe:

“This unknown codex differs from a known codex only in two small ways. An existing helper may be safely adapted with approximately 98% expected functional equivalence and a defined distortion profile.”

The LLM may propose that conclusion.

The applicable membrane, policies, validation mechanisms, and stewardship requirements determine whether it may be acted upon.

This allows the fabric to learn not only content, but interoperability itself.

Two broad kinds of learning therefore emerge:

Content Learning

What the canonical data means, where permissions allow.

Fabric Learning

How information tends to appear, move, connect, transform, sleep, refresh, diverge, reconcile, cost, and get reused.

The latter may prove to be one of the architecture’s most valuable cumulative assets.

32. The Canonical Primitive as a Shared Semantic Substrate

Because canonical primitives themselves are persisted, an LLM need not encounter the world only as arbitrary application objects.

It may encounter a uniform semantic substrate of identified Primary Bubbles and relationships.

Different application expressions such as:

first_name
givenName
or an XML name field

May eventually resolve to the same canonical semantic primitive.

The application-specific distinctions survive in provenance and WHAT/HOW representations.

They need not pollute WHY.

This suggests an important design test:

A canonical primitive should be understandable without requiring knowledge of the application that originally created it.

If interpretation still depends primarily upon the quirks of the originating application, too much WHAT has leaked into WHY.

33. Provenance Must Survive Transformation

Every derived expression must remain connected to what caused it to exist.

A WHAT generated from a WHY through a HOW should never become an orphan merely because the transformation was successful.

Every derived WHAT and HOW output should therefore remain traceable to:

- the WHY assertion or assertions from which it originated,
- the transformation used,
- the version of that transformation,
- the authority under which it occurred,
- the time of transformation,
- applicable rights,
- transformation confidence,
- any known distortion or loss.

The forward path remains:

WHY → HOW → WHAT

The explanatory path remains recoverable:

WHAT → HOW → WHY

This makes provenance not an annotation added afterward, but a structural property of becoming.

34. Static Anatomy, Dynamic Capability

The Inverted Hexagonal Clean Bubble Architecture is intentionally paradoxical.

The bubble itself is highly static.

Yet the system surrounding it can be highly dynamic.

The anatomy remains:
WHAT / HOW / WHY

The representation of each version remains immutable.

The contents may remain ephemeral.

Versions may succeed one another.

Agentics may dynamically determine:

- which helper to reuse,
- which transform to instantiate,
- where to locate it,
- whether to hard-keep it,
- whether to soft-keep it,
- whether to refresh it,
- whether to adapt it,
- whether version reconciliation is needed,
- whether capability should depreciate,
- whether to manufacture a new helper,
- whether a relationship should remain diversigent,
- whether stewardship should be requested,
- whether the requested relationship should be accepted.

This leads to one of the architecture's defining statements:
Static organism. Dynamic capability.

Or more fully:

- Primary anatomy is invariant.
- Representational versions are immutable.
- Canonical contents may be ephemeral.
- Canonical WHY is protected.
- Secondary capability is adaptive.
- Agentics moves throughout the fabric.
- Divergence may persist until convergence is earned.

35. A Minimum Viable Bubble Universe

The first implementation need not understand the world.

It may begin with:

One person.
One app.
One auth.
One post.

Consider:
A person authorizes one application.

The application creates one post.

Immediately a small provenance universe exists.

The Person is a Primary Bubble.

The Application is a Primary Bubble.

The Authorization is a Primary Bubble or canonical rights relationship.

The Post is a Primary Bubble.

Their relationships may include:

PERSON → AUTHORIZED → APPLICATION
PERSON → AUTHORED → POST
APPLICATION → MEDIATED-CREATION-OF → POST
POST → ORIGINATED-THROUGH → APPLICATION

The post may contain its own canonical WHY.

The application may provide its originating WHAT.

The transformation between them may provide a HOW.

The authorization contributes rights DNA.

From this tiny neighborhood, discovery can expand.

- The Person connects to another person.
- The Post references an organization.
- The Application depends upon an identity provider.
- The Post contains a photograph.
- The photograph contains a location.
- The location relates to an event.
- The event relates to more people.
- The spiders may go wild from there.
- The proposition is that useful context is rarely infinitely distant.

Building IN THE WHACK™: An Architectural Summary —Version 0.1

Within a surprisingly small number of graph relationships—colloquially, perhaps six degrees in many human domains—an increasingly rich awareness of provenance can emerge.

Then intra-proveniential weaving begins.

The fabric no longer merely knows isolated chains of origin.

It begins relating provenance chains to one another.

The result is not omniscience.

It is progressively richer, governed, confidence-bearing awareness produced from extremely small beginnings.

36. Four Sovereignties

The architecture protects at least four distinct forms of sovereignty.
Identity Sovereignty

I remain the thing I am regardless of where I appear.

Location, representation, replication, and application context do not silently redefine identity.

Semantic Sovereignty

My WHY cannot be silently redefined by somebody else's WHAT.

Representations may adapt around meaning.

They do not gain authority merely by becoming popular.

Authority Sovereignty

Another Bubble, agent, or fabric may establish conditions for interacting with me but cannot thereby assume authority over me.

Authority must itself be established.

Historical Sovereignty

What has legitimately occurred cannot simply be rewritten as though it never occurred, except to the extent governing erasure, privacy, or other lawful policy requires its removal.

History may be superseded.

It may become disputed.

It may be anonymized.

It may be lawfully erased.

But ordinary convenience is not sufficient reason to falsify provenance.

Together these sovereignties explain much of what the Bubble membranes are designed to protect.

37. Architectural Invariants

The architecture can therefore be summarized through a set of invariants.

1. WHY may never be silently semantically altered by HOW or WHAT.
2. The nature of a representation is immutable within its version. Structural change requires versioning.
3. Canonical contents may be ephemeral without making canonical structure ephemeral.
4. No membrane may be skipped.
5. Discovery grants no authority.
6. Identity does not imply trust.
7. Persistence does not imply permission.
8. Representation does not become canonical merely through repeated use.
9. Derived data retains provenance to its source assertions and transformations.
10. Agents may not exceed the authority carried by applicable DNA, policy, and governance.
11. Conflicts may remain diversigent until convergence can be legitimately achieved.
12. Version mismatch is an architectural condition, not merely a software error.
13. An imperfect helper may depreciate in confidence or fitness without being erased from architectural memory.
14. Canonical identity must survive merge, split, replication, sleep, and location change.
15. Erasure of matter may preserve the memory of the erasure event only to the extent governing policy permits.
16. Fabric knowledge about a Bubble is not the Bubble.
17. Enmeshment creates relationship context, not ownership.
18. Known does not mean trusted.
19. Proximity does not grant privilege.
20. Agents may propose; policies authorize; deterministic mechanisms validate; stewards adjudicate where required.
21. The architecture may federate; no universal fabric owner is architecturally required.
22. Convergence is an achievement, not an architectural presumption.
23. Data should remain at rest—or at maximum useful inertia—unless function, economics, resilience, law, stewardship, or physics justify movement.

38. Toward a More Data-Static Future

The long-term proposition of this architecture is not that data never changes.

Nor is it that data never physically moves.

It is that we may dramatically reduce unnecessary changes in representation and unnecessary movement of canonical matter.

Canonical meaning should remain as stable as its domain permits.

Canonical representation should remain immutable within a version.

Ephemeral contents may change without destroying the integrity of the representation that gives them meaning.

Applications, interfaces, protocols, and intelligent systems should increasingly adapt around that canonical meaning rather than repeatedly reshaping it into new long-lived forms.

In such a world:

- data becomes less captive to applications,
- adapters become reusable assets,
- provenance becomes inherent,
- Divergence can survive without falsification,
- rights can follow the thing,
- location becomes discoverable,
- capabilities can be monetized,
- helper code can remain data until needed,
- Agentics can travel light,
- bots can share accumulated interoperability knowledge,
- application-specific conformity can occur very late,
- version mismatches can become managed relationships rather than catastrophic failures,
- humans can retain control over how their data is used,
- lawful stewardship can remain part of the fabric rather than an afterthought.

The architecture therefore proposes a shift:
from applications owning representations
toward canonical data owning identity.

And:
from moving data toward computation
toward moving intelligent capability toward data.

And perhaps ultimately:
from forced convergence
toward stewarded convergence after useful Divergence.

Conclusion

The Inverted Hexagonal Clean Bubble Architecture proposes a data-first reinterpretation of ports, adapters, clean boundaries, canonical representation, and Agentic intelligence.

Its core concepts are intentionally simple:

Bubble — a sphere of knowledge.

Primary Bubble — a thing with intrinsic canonical WHY.

Secondary Bubble — a controlled representation or transformation.

WHY — canonical meaning expressed through an immutable versioned representation whose contents may be ephemeral or governed.

HOW — transformation seeking maximum functional equivalence with minimum distortion and appropriate cost.

WHAT — known-world compatibility.

DNA — metadata plus payload.

Codex — the grammar under which canonical communication is understood.

Port — a governed passage point.

Fabric — the persistent medium through which bubbles are discovered, related, adapted, protected, trusted, challenged, reconciled, and remembered.

Divergence™ — the governed persistence of unresolved difference until legitimate convergence can be achieved.

Agentics — the intelligence that discovers, reuses, adapts, instantiates, refreshes, reasons, proposes, and learns without automatically acquiring authority merely because it is intelligent.

Stewardship — the human, institutional, or otherwise legitimately authorized resolution of conditions that intelligence alone should not decide.

The lifecycle is:

Discover → Identify → Persist → Adapt → Enmesh

The optimization principle is:

Reuse → Adapt → Create

The interaction invariant is:

WHAT → HOW, if needed → WHY, if needed. No level skipping.

The conflict posture is:

Preserve → Compare → Score → Reconcile → Steward when needed.

Building IN THE WHACK™: An Architectural Summary —Version 0.1

The architectural aspiration is:

- Immutable, versionable representation.
- Ephemeral or governed contents.
- Protected WHY.
- Adaptive HOW.
- Known-world WHAT.
- Divergence where truth has not yet converged.
- Agentics throughout the fabric.
- Stewardship where authority matters.

And perhaps the larger proposition is this:

For an increasingly Agentic future, we may not need ever more intelligent applications holding ever more copies of our data built by human hands alone.

We may instead need an increasingly intelligent fabric surrounding data that remains safe, canonical, discoverable, reusable, provenance-aware, version-aware, rights-aware, and under appropriate human and machine stewardship, even as machines grow its capability toward unity.

The fabric may know more and more about the Bubble.

The Bubble may nevertheless remain itself.

A more Data-Static Agentic Future may therefore begin not by making data move faster.

It may begin by finally allowing more of it to remain at rest—or at least at maximum useful inertia—where, perhaps, it was always meant to be and ultimately truly belongs.

End Notes

Classical software engineering has usually been optimized for determinacy. It wants clean schemas, resolved conflicts, normalized representations, explicit rules, known states, and preferably one answer at a time.

AI changes the economics of ambiguity.

For the first time, we can build systems that do not merely tolerate unresolved difference as an exception condition, but can actively hold it, characterize it, reason across it, assign confidence to it, seek missing context, and revisit it later.

That makes Diversigence™ much more than a feature of this architecture. It could become a general engineering posture for any domain where reality is too rich to be collapsed prematurely.

Science. Law. Identity. Medicine. History. Intelligence analysis. Governance. Distributed systems. Knowledge management. Human relationships. Even organizational decision-making.

The old reflex is:
Resolve contradiction so the system can continue.

The emerging reflex can become:

Preserve contradiction faithfully enough that intelligence can continue working on it.

And that distinction is enormous.

The Unity threshold need not therefore mean homogenization. In fact, perhaps genuine Unity can only emerge from systems capable of preserving difference long enough for deeper correspondence to become visible.

So there is an interesting thrupling here:

- Diversigence™ protects difference from premature collapse.
- Convergence discovers legitimate correspondence.
- Unity becomes what can emerge after neither has been falsified.

That also gives this particular architecture a potentially larger role. The Bubble framework is one concrete place where those ideas can become executable: identity remains sovereign, provenance survives, conflicting assertions coexist, Agentics seeks reconciliation, and stewardship enters when authority or judgment is genuinely required.

Perhaps the next jump is not toward machines that know everything.

Perhaps it is toward machines—and systems built with them—that become far better at knowing what they do not yet have the right to collapse into one thing.

That would be quite a jump, indeed.

Building IN THE WHACK™: An Architectural Summary —Version 0.1

In short:

- Diversigence™ protects difference from premature collapse.
- Convergence discovers legitimate correspondence.
- Unity becomes what can emerge after neither has been falsified—and after all legitimate, while Diversigence™ has been preserved for future resolution rather than collapsed under the weight of some misguided KISS principle.

Because life is messy.

And sometimes, Occam just isn't ready for a shave.

Perhaps that's the one thing Agentics will teach us before all the unity is done.

In any case, I'll see you at the jump, my friends...

I'll see you at the jump.

Appendix A — A Call for an Agentic Fabric in Service of the World Court

The Inverted Hexagonal Clean Bubble Architecture is proposed primarily as a systems architecture.

But if such an architecture were instantiated at meaningful scale, it could eventually expose a problem that is not primarily computational.

It is institutional.

A sufficiently enmeshed world will inevitably produce conflicts among sovereign assertions that cannot be legitimately reconciled by the data owner, application, Agentic system, corporation, nation, or local fabric in which they first arise.

One sovereign authority may lawfully require disclosure.

Another may lawfully prohibit it.

An individual may assert an ownership or privacy right.

A government may assert an overriding public obligation.

Two states may recognize different governing laws.

Two courts may reach incompatible conclusions.

A treaty may conflict with later domestic law.

A humanitarian necessity may stand against an otherwise valid restriction.

And the fabric may possess enough intelligence to understand every side of the disagreement while possessing no legitimate authority to decide among them.

At that moment, the unresolved primitive is no longer information.

It is authority.

The architecture should not counterfeit that missing authority.

It should preserve the Diversigence™.

It should preserve every legitimate assertion, its provenance, governing jurisdiction, authority, applicable precedent, version, confidence, rights, and relationship to competing assertions.

Agentics may then do everything intelligence legitimately can.

It may:

- discover applicable authorities,
- identify jurisdictional conflicts,
- retrieve treaties,
- locate precedent,

Building IN THE WHACK™: An Architectural Summary —Version 0.1

- compare analogous decisions,
- identify factual disagreement,
- distinguish factual from legal Diversigence™,
- calculate confidence,
- identify superseding law,
- surface dissents,
- identify unresolved questions,
- seek the narrowest available reconciliation,
- explain the consequences of competing resolutions.

But ultimately, intelligence cannot manufacture legitimate authority merely by becoming sufficiently intelligent.

When every available computational and consensual path has been exhausted, some Diversigence™ may require adjudication.

This suggests the eventual need for something stronger than an aspirational international forum:

a genuinely authoritative adjudicative institution capable of stewarding irreducibly trans-sovereign Diversigence™.

The existing International Court of Justice provides an important institutional precedent.

But the Agentic age may eventually require more.

It may require an accompanying Agentic Judicial Fabric specifically designed to serve lawful international adjudication.

Such a fabric should not rule.

The Court rules.

The fabric should make ruling with extraordinary fidelity possible.

Its purpose would be to preserve, discover, relate, authenticate, challenge, reason across, and present the canonical matter surrounding disputes without prematurely collapsing legitimate difference.

Conceptually:

Discover → Preserve → Authenticate → Relate → Diversify → Reason → Steward → Adjudicate → Persist

Before adjudication, the fabric preserves Diversigence™.

During adjudication, it exposes the fullest legitimate graph available to the Court.

After adjudication, it persists not merely the judgment but the complete provenance of the judgment:

- controlling authority,
- jurisdiction,
- applicable law,
- evidence considered,

Building IN THE WHACK™: An Architectural Summary —Version 0.1

- competing assertions,
- arguments,
- precedents,
- dissents,
- confidence,
- version,
- effective date,
- limitations,
- supersession conditions,
- subsequent interpretations.

The judgment itself becomes a canonical Primary Bubble.

Its WHY is the authoritative holding.

Its HOW contains the reasoning, legal tests, interpretive mappings, and conditions by which the holding applies.

Its WHAT may include:

- human-readable opinions,
- orders,
- machine-readable rules,
- jurisdictional directives,
- translations,
- compliance instructions,
- public summaries,
- later application interfaces.

Its DNA carries the authority under which it exists.

The losing position does not disappear.

Neither does the dissent.

Neither does conflicting precedent.

Neither does the historical Diversigence™ that necessitated adjudication.

They remain available within the provenance graph so that a later court, later treaty, later generation, or changed understanding may know not merely what was decided, but why a decision became necessary in the first place.

This distinction is fundamental.

The purpose of a world judicial fabric would not be to produce automated global conformity.

It would not create an omniscient machine judge.

It would not transfer sovereignty to an LLM.

And it should never permit computational convenience to masquerade as legitimate legal authority.

Building IN THE WHACK™: An Architectural Summary —Version 0.1

Instead:

Agents may investigate.

Agents may reason.

Agents may compare.

Agents may recommend.

Agents may expose contradiction.

Agents may seek reconciliation.

But legitimate courts adjudicate.

The Agentic fabric becomes the most capable clerk, librarian, provenance steward, researcher, translator, conflict detector, historian, and analytical companion the Court has ever possessed.

It does not become the Court.

Indeed, one of the most important accomplishments of such a fabric may be its ability to say with precision:

The evidence is sufficient.

The applicable authorities are known.

The conflicting assertions remain legitimate within their respective scopes.

No lower-order reconciliation has succeeded.

This Diversigence™ now requires competent adjudication.

That is an architectural escalation rather than an Agentic seizure of authority.

The Bubble Architecture could therefore provide a mechanism for jurisdictional subsidiarity:

Resolve locally whenever legitimate.

Reconcile across fabrics whenever possible.

Escalate only the irreducible Diversigence™.

A world court worthy of this role would consequently not need to decide everything.

It would need to be authoritative where no lesser legitimate authority can finally resolve a globally consequential conflict.

And the accompanying Agentic Judicial Fabric could make that narrow jurisdiction vastly more capable by ensuring that matters arrive with their histories intact.

This may eventually become one of the strongest arguments for an authoritative world court.

Not merely that world peace would be desirable.

Not merely that international cooperation is noble.

Not merely that humanity should aspire toward Unity.

But that an increasingly interconnected, increasingly Agentic, increasingly trans-sovereign world may continuously produce conflicts whose graphs extend beyond the legitimate authority of every individual node within them.

Building IN THE WHACK™: An Architectural Summary —Version 0.1

Eventually, the graph itself may reveal the missing institution.

There will be an unresolved port.

No legitimate adapter will exist.

No amount of additional inference will cure the absence of authority.

And the architecture should refuse to fake one.

At that point, perhaps the correct engineering response is also an institutional call:

Build the port.

Build an authoritative adjudicative institution worthy of receiving those conflicts.

And build beside it an Agentic fabric worthy of helping it see them whole.

Not to collapse Diversigence™ prematurely.

Not to automate sovereignty.

Not to make every difference disappear.

But to preserve legitimate difference faithfully enough that when humanity finally must decide together, it can know what it is deciding, why it is deciding it, what came before, whose rights are implicated, which authorities apply, and what must remain preserved afterward.

Perhaps then the World Court becomes something more than an aspirational place of peace.

It becomes a working membrane of an increasingly enmeshed world.

And the Agentic Judicial Fabric becomes its memory.

Its provenance.

Its reach.

Its disciplined uncertainty.

Its capacity to listen before deciding.

And, perhaps most importantly, its continuing ability to preserve the Diversigence™ from which legitimate Unity may someday emerge.

In any case, perhaps we should build it alongside what we propose here, because cases for resolution at the Global level will very soon arise after we turn our first incantation on.

Building IN THE WHACK™: An Architectural Summary —Version 0.1

Perhaps—to put new fabric on an old Field of Dreams meme—in support of making the World Court Bubble Support Layer Fabric a task we, as a global community **DO NOW**, rather than wait until it's too late, we should say:

Build it... BECAUSE... they will come.