C. IFII's Engineering Program and Preliminary Development Plan

Table of Contents

C. IFII's Engineering Program and Preliminary Development Plan

READER ORIENTATION:

This document is a programmatic deep dive accompanying IFII's vision statement. Its purpose is not to introduce IFII at the broadest level, but to show how IFII's public-good vision can be developed into a disciplined research, engineering, and validation program under explicit boundary conditions. The document explains how selected source-informed concepts are handled through terminology control and source-fidelity review, how the General Theory of Formation (GTF) becomes researchable through the Constructive Formation Model (CFM), and how the Accountability Formation Environment (AFE) can serve as a non-sentient research instrument for testing formation-accountability design. Readers new to IFII may wish to first consult IFII's vision statement, which provides the broader mission, public-good framing, and citizen-engineer orientation.

1. Field-Informed Intelligence – An Engineering Approach for the 21st Century

1.1 Engineering Premise: Technology Builds Worlds

Technology builds worlds. Every intelligent system shapes what appears, what is attended to, what is valued, and what becomes actionable. For this reason, responsible engineering can no longer ask only whether a system is performant, scalable, or efficient. It must also ask what kind of body-world architecture the system creates: which formations it stabilizes, which causes it hides, which actions it invites, and whether humans remain able to recognize, challenge, revise, and release what the system produces. Here, "body-world architecture" means the coupled pattern of interface, attention, action, feedback, dependency, and environment through which a system shapes how users and institutions experience and act within a world.

Field-informed intelligence is IFII's engineering paradigm for this question. It names intelligence that remains in relation to context, whole, and consequence. In human-made systems, this means designing architectures, feedback loops, validation methods, and governance structures that support coherence rather than fragmentation, recognition rather than capture, responsiveness rather than extraction, and meaningful participation rather than isolated optimization. In practical citizen-engineer language, IFII calls this orientation Reality Alignment: designing and acting in ways that preserve context, consequence, human agency, reality-status awareness, and the possibility of revision or release.

1.2 IFII's Applied Research Architecture

C. IFII's Engineering Program and Preliminary Development Plan

IFII draws from selected contemplative, philosophical, phenomenological, systems, and technical sources, including Tibetan Buddhist and Bön materials where relevant, as comparative sources for questions about appearance, attention, fixation, recognition, context, non-reification, and release. Dzogchen / Great Perfection materials are part of the initial source corpus, but IFII does not claim to represent Dzogchen, define Dharma, prove contemplative accounts through technology, reduce them to computation, or turn realization into an engineering object. The research task is bounded: identify selected structural insights, reformulate them into candidate concepts, design constraints, hypotheses, requirements, and validation questions, build bounded prototypes, and subject all outputs to source review, interdisciplinary critique, verification, validation, challenge, and revision.

IFII's General Theory of Formation (GTF) asks how appearances, mental/model objects, and actualized formations arise, stabilize, perpetuate, transform, dissolve, and become available for recognition or non-reification. It begins with candidate primitives such as field, geometry, boundary condition, frequency, resonance, formation, memory, recognition, and responsiveness. These are not treated as slogans, finished truths, or source claims. They are candidate research objects profiled through source-corpus meaning, phenomenological meaning, scientific analogue, computational analogue, failure mode, requirement seed, and prototype exposure.

The Constructive Formation Model (CFM) is the method that makes the GTF researchable and engineerable. It reformulates candidate primitives into primitive profiles, coupling maps, formation axioms, derived claims, hypothetico-deductive (H-D) hypotheses, requirements, architecture allocations, prototypes, verification artifacts, validation protocols, and learning-return records. In this sense, the CFM is a theory-to-engineering bridge: it turns source-informed inquiry into method, method into requirements, requirements into prototype behavior, and prototype evidence back into refined understanding.

Layer	Function	Primary Artifact	Governance Question
IFII	Institutional R&D and public-good governance container	Research program, roles, review gates, public documentation	Who is accountable for source fidelity, scope, safety, and dissemination?
GTF	Long-term theory-construction program informed by bounded source dialogue and cross-domain testing	Candidate primitives, formation axioms, formation metrics	Is the proposed principle source-faithful, coherent, necessary, and testable?

Layer	Function	Primary Artifact	Governance Question
CFM	Source-fidelity-governed reformulation process from bounded source corpus and source review to candidate axioms, derived claims, hypotheses, requirements, and validation protocols	Primitive registry, axiom handover, RSM, V&V records	Is the chain of custody from source to requirement auditable?
CFM Automation Project	AI-assisted, human-governed infrastructure that scales the CFM	Source repository, workflow, traceability graph, versioning, review gates	Does automation support human judgment without replacing it?
AFE	Non-sentient research instrument and reference architecture for testing formation-accountability design	AFE reference core, vertical slices, validation evidence, learning return	Did the prototype make formations visible, contestable, responsive, and releasable?

1.3 Formation Accountability and the Black-Box Problem

This architecture directly addresses the black-box problem in intelligent systems. A black box can be useful as a modeling abstraction when only inputs and outputs are relevant. It becomes dangerous when used as an accountability model for self-regulating technologies that shape human attention, judgment, institutions, and social reality. If a system's internal formation logic is inaccessible, then responsibility becomes diffused: designers cannot fully explain what was built, users cannot understand what they are entering, reviewers cannot audit causal pathways, and institutions can treat opacity as inevitability. IFII's response is formation accountability: every significant output, model, recommendation, or interface event should be traceable to its sources, assumptions, boundary conditions, requirements, risk controls, and validation evidence.

The AFE therefore does not implement contemplative-realization or attainment-related concepts. In the public engineering register, the relevant principle is a conventional formation-and-release lifecycle: system-generated outputs and interface states should disclose their sources and conditions, remain distinguishable from their referents, remain provisional, and remain capable of revision, release, or retirement. This is a source-informed analogy under strict guardrails, not a claim of realization. The AFE is not an enlightened machine, a sentient agent, a source-fidelity reviewer, or a metaphysical proof. It is a disciplined attempt to build non-reifying

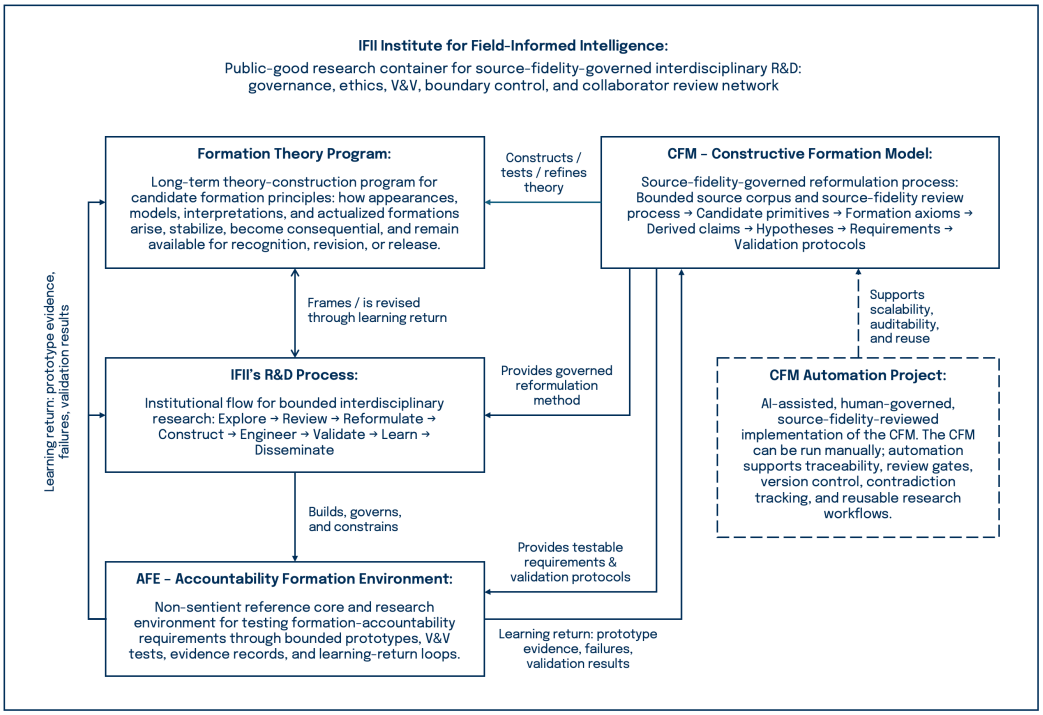
formation accountability into technical architecture: the system should not conceal causes, confuse model-output with referent, reify appearances as final reality, capture attention, block revision, or prevent release.

1.4 Dzogchen Source Concepts and Terminology-Control Policy

The key policy is terminological discipline. IFII should preserve source terms only where they are needed, reformulate them consistently, and explicitly distinguish five levels of use: source-facing meaning, English bridge term, engineering analogue, implementation requirement, and guardrail. This protects the project from two opposite errors: flattening contemplative traditions into generic design vocabulary, or over-spiritualizing engineering until the work becomes untestable, unsafe, or misleading.

1.5 IFII's Core Engineering Thesis

Every system-generated appearance, model, recommendation, interface state, prototype event, or actualized formation produced by an IFII-developed system must disclose its conditions, remain distinguishable from its referent, remain contestable, serve a bounded purpose, and be capable of revision or release. The AFE is the research instrument for testing whether such an architecture is possible. The CFM is the process that derives the requirements. The GTF is the theory that proposes the formation principles. IFII is the accountable institutional container that governs the whole effort as a public-good research program:



1.6 IFII's Call to Action

IFII's technical work ultimately serves a broader public-good and citizen-engineer mission. The goal is not only to build an internal research system, but to develop standards, methods, reference architectures, and validation practices that help scientists, engineers, founders, funders, artists, educators, and institutions build technologies that preserve coherence, responsiveness, human agency, and reality-status awareness. The AFE is therefore not only a prototype target; it is also a test case for a wider ecosystem of field-informed engineering practice.

2. The Constructive Formation Model: A Primitive-to-Prototype Process for Field-Informed Intelligence

In IFII's methodological architecture, technology is both research instrument and governance substrate. The development of the General Theory of Formation (GTF) proceeds through a hybrid Constructive-Deductive approach that operates in tandem with the design, verification, and validation of bounded formation-accountability systems.

The program employs a Constructive Formation Model (CFM): a primitive-first, axiom-led, prototype-exposed, source-fidelity-protected, and verification-and-validation-governed process. The CFM no longer begins with isolated empirical primitives or with already-known requirements. It begins with a domain boundary, a bounded source corpus, and a candidate primitive constellation that can be profiled, coupled, formalized, engineered, tested, and revised. "Bounded source corpus" means the provenance-controlled body of materials, interpretations, and review decisions accepted as source material for a given research cycle; it does not mean that IFII claims authority to define a Dharma canon.

The model integrates four methodological movements:

1. **Orientation and primitive discovery:** Define the domain, source boundary, scientific corpus, use-case context, and candidate primitive constellation.
2. **Constructive formation:** Profile candidate primitives through multiple lenses, map how they co-condition one another, derive operational behavior, geometry, metrics, and formation axioms.
3. **Deductive engineering:** Translate formation axioms into derived claims, falsifiable hypotheses, requirements, architecture allocations, design patterns, prototypes, and verification artifacts.

4. **Systemic validation and learning return:** Validate prototypes against the original intent, stress-test them against rival explanations, and feed the results back into the primitive registry, axiom versioning, and requirements baseline.

This recursive integration ensures that theory and practice remain mutually corrective. Candidate theory is tested through the success, failure, and correction of prototypes; technological design becomes a means of probing and refining the relational principles the theory proposes, not proof of metaphysical truth.

The formal external name of the model is the Constructive Formation Model. Its operational subtitle is "A Primitive-to-Prototype Process for Field-Informed Intelligence." Its key artifacts are the primitive registry, the formation axiom handover, the Requirements Specification Matrix (RSM), architecture allocation records, verification artifacts, validation protocols, and learning-return records.

These tracks are integrated as follows:

Dimension	Verification Question	Validation Question	Feedback Purpose
Theoretical (GTF) / Constructive	Did we derive the primitive profile, coupling map, geometry, metric, and formation axiom correctly?	Did we build the right theory? Does the constructed model align with the source corpus, phenomenological data, scientific analogues, computational analogues, and prototype-exposable behavior?	Did we derive the primitive profile, coupling map, geometry, metric, and formation axiom correctly?
Primitive Specification / Governance	Did the candidate primitive pass its status decision and traceability checks?	Does the primitive registry preserve source fidelity while avoiding category errors and unsupported metaphor?	Revise the primitive profile, downgrade metaphorical material, add missing lenses, or reject unsuitable candidates.
Technological (AFE) / Deductive	Did we build the system right? Are requirements, architecture allocations, component tests,	Did we build the right system? Does the system output satisfy the intent of the formation axioms and	Refine requirements, hardware and software constraints, risk controls, design

Dimension	Verification Question	Validation Question	Feedback Purpose
	simulations, and risk controls implemented as specified?	demonstrate relational coherence under stress conditions?	patterns, and implementation architecture.
Learning Return / Reflexive Loop	Did the evidence clearly support, weaken, or falsify the relevant claim, hypothesis, or requirement?	Does the learning improve the GTF, the AFE, and the governance of future prototypes?	Update the primitive registry, formation axiom set, RSM, validation metrics, and future research agenda.

3. The Axiomatization Pipeline

The Constructive Formation Model integrates the development of the General Theory of Formation (GTF) with systems-engineering and design-control practices. Standard systems engineering assumes that requirements are known. In foundational research, however, the relevant candidate principles, primitives, and requirements must first be constructed and justified. The CFM therefore maps the constructive derivation of a formation axiom from a candidate primitive constellation to the empirical validation of the resultant system behavior.

The pathway is: bounded source corpus and related research corpus → candidate primitives → seven-lens primitive profiles → primitive status decisions → primitive coupling maps → operational behavior → formal or topological construction → metrics and state variables → formation axioms → derived claims → H-D hypotheses → requirements → architecture allocation → prototypes → verification artifacts → validation protocols → learning return.

A formation axiom is not an arbitrary definition. It is a constructed candidate principle that describes how appearances, mental/model objects, and actualized formations arise, stabilize, perpetuate, transform, dissolve, and become available for recognition, non-reification, or revision. Each formation axiom must be traceable backward to its primitive constellation and forward to claims, hypotheses, requirements, design patterns, and validation evidence.

3.1 The Double Epistemic Loop: Mitigating Tautology

The entire pipeline is rooted in disciplined research method. However, relying solely on a traditional axiomatic-deductive system carries the philosophical risk of becoming a closed, tautological system that verifies its own definitions rather than the phenomena

and use contexts it claims to address. This risk is especially acute in AI architecture: if a system begins with a narrow definition of intelligence, builds an artifact optimized for that definition, and then treats the artifact's success as proof of the definition, the process confirms itself rather than the world it claims to describe.

IFII mitigates this risk through a double epistemic loop:

1. **Loop 1 – Constructive Validation:** Test the candidate primitive, coupling map, formal representation, metric, and formation axiom against the bounded source corpus, phenomenological data where appropriate, scientific analogues, computational analogues, failure modes, rival explanations, and prototype-exposable behavior. This loop governs the idea. Its output is an accepted, revised, downgraded, or rejected formation axiom. Its standard is source fidelity, coherence, necessity, and testability.
2. **Loop 2 – Systemic Verification and Validation:** Test the engineered system against the formation Axiom, derived claims, hypotheses, requirements, architecture allocations, and validation protocols. This loop governs the artifact. Its output is a verified, validated, corrected, or rejected prototype. Its standard is compliance, effectiveness, robustness, and empirical learning.

The system establishes empirical contact through design hypotheses. These hypotheses link candidate formation principles to observable, measurable, or prototype-exposable behavior. By requiring implemented properties to generate reviewable evidence under defined conditions, V&V results feed back into axiomatic coherence. The model thereby prevents the theory from becoming disconnected, purely interpretive, or self-confirming.

3.2 The Primitive Specification Layer and the Primitive Registry

The candidate primitives are drawn from multiple domains: selected source corpus, phenomenological reports where appropriate, physical and scientific analogues, computational analogues, observed failure modes in current technology, and prototype-exposable behavior.

The first substantive step is therefore not an isolated empirical primitive. It is a candidate primitive / primitive constellation. A candidate primitive is a recurring, provisionally foundational formation concept identified across source material, phenomenological reports where appropriate, scientific analogy, computational failure modes, and prototype-exposable behavior. A primitive constellation is a mutually conditioning set of such primitives. For IFII, the initial working constellation includes field, geometry, boundary condition, frequency, resonance, formation, memory, and recognition.

C. IFII's Engineering Program and Preliminary Development Plan

The term *constellation* matters because IFII is unlikely to derive the GTF from a single primitive. The core terms co-condition one another. For example:

- field without boundary remains open potential;
- boundary without frequency remains static constraint;
- frequency within boundary produces resonance;
- resonance stabilizes formation;
- repeated formation creates memory;
- recognition releases reification

The primitive registry is the structured artifact that stores these profiles, decisions, couplings, and traceability links. It makes the work scalable, auditable, interdisciplinary, reusable, and reviewable by different audiences. It also protects the project from category errors: poetic metaphor must not be mistaken for engineering principle, and computational analogy must not be mistaken for source meaning.

For each candidate primitive, IFII creates a Seven-Lens Primitive Profile:

No.	Lens	Guiding Question
1	Source Corpus / Source Meaning	What does the term mean within the bounded, review-governed source corpus, and what limits must be preserved before any engineering analogue is proposed?
2	Phenomenological	How is the primitive encountered in lived experience, contemplative observation, perception, embodiment, or attention?
3	Scientific	What is the closest physical, biological, mathematical, or systems-theory analogue, and where does the analogy break down?
4	Computational	What would this mean in software, AI, information architecture, agent design, memory, interface, or state-space behavior?
5	Failure Mode	Which current technological limitation does this primitive explain, expose, or help solve?
6	Requirement Seed	What preliminary "shall" statement or design obligation could eventually be derived from it?
7	Prototype Exposure	Which experiment, simulation, demonstrator, visual interface, hardware setup, or field-sensing tool could make it observable?

3.3 Preliminary Failure Mode Taxonomy

Current AI architectures, including transformer-based systems, deliver powerful selective focus over token-to-token relations. But many real-world problems require more than local relevance: they need stable contextual spaces that persist across time, tasks, and modalities; explicit rules and constraints that govern long-horizon coherence; and relational dynamics that integrate multiple processes without drift. IFII's position is pragmatic: before proposing alternatives or additions, the project must substantiate where current systems actually fail, why they fail, whether scale alone fixes those failures, and how compute, energy, and hardware constraints amplify the problem.

This scoping domain turns scattered anecdotes into a structured evidence base – a failure taxonomy, missing processes, scaling limits, efficiency analysis, and hardware constraints – so that subsequent IFII systems design initiatives are grounded in substantiated limitations, not in anecdote or rejection of current AI.

A preliminary literature review of 40 papers (2022-2025) focusing on AI models and their performance on long-horizon, context-dependent tasks suggests a structured taxonomy of failures that motivates focused investigation.

Failure Modes Identified: The analysis identified 11 failure modes that may compromise core model capabilities such as context retention, computational accuracy, and systematic reasoning under defined conditions.

Severity Profile: Within the scoped review, these modes appeared to form a consequential risk profile. The severity and frequency ratings assigned to these failures break down as follows:

- **High Severity (5 modes):** Context Drift, Brittle Rule Switching, Inability to Perform Random Access, Lost-in-the-Middle, and Bandwidth Constraints.
- **Moderate to High Severity (2 modes):** Mentation Entropy/Dispersion and Instruction Drift.
- **Moderate Severity (4 modes):** Recency/Primacy Bias, Numerical Fragility, Action Repetition/Illegal Actions, and Logit Explosion/Training Instability.

Safety and Effectiveness Implication: The preliminary classification suggests that the observed instability is consequential for tasks requiring sustained context and systematic reasoning. Among the included studies, IFII did not identify clearly low-risk modes within the scoped failure set. This should be treated as an initial signal for focused investigation, not as a final general claim about all AI architectures or all possible scaling regimes.

3.4 The Construction Grid and Traceability

The construction grid operates from Step 0 to Step 17. The grid creates an unbroken chain of custody from domain and source boundary, through primitive discovery and constructive formation, to engineering requirements, architecture allocation, prototype verification, validation, and learning return. This structure ensures that every component of the implemented system contributes demonstrably to supporting, challenging, testing, falsifying, or advancing GTF candidate claims.

C. IFII's Engineering Program and Preliminary Development Plan

Phase	Step	Element (What?)	Description (How?)	Linkage and Purpose (Why?)
I. Orientation	0	Domain and Source Boundary	Define the domain: AI agent, formation chamber, field-sensing tool, analog formation space, or another bounded research object. Identify source corpus, scientific corpus, and use-case context.	Prevents overgeneralization and clarifies what the model is trying to construct.
II. Primitive Discovery	1	Candidate Primitive Constellation	Identify candidate primitives such as field, geometry, boundary condition, frequency, resonance, formation, memory, and recognition.	Starts with candidate formation primitives rather than assumed laws.
	2	Seven-Lens Primitive Profile	Profile each primitive through source-corpus, phenomenological, scientific, computational, failure-mode, requirement-seed, and prototype-exposure lenses.	Turns source-informed and theoretical terms into controlled research objects.
	3	Primitive Status Decision	Decide whether a primitive is source-reviewed, phenomenological, scientific, computational, metaphorical, or rejected.	Prevents metaphor, analogy, or interpretation from being mistaken for source truth or engineering principle.
III. Constructive Formation	4	Operational Behavior	Define what the primitive does (primitive function): constrains, opens, resonates, stabilizes, remembers, recognizes, releases, dissolves, or forms.	Converts meaning into behavior required for operationalization.
	5	Primitive Coupling Map	Define how primitives co-condition each other, e.g. boundary + frequency → resonance → formation → memory → recognition.	Adds interdependence explicitly into the process.
	6	Geometric / Topological Construction	Derive the implied structure: field topology, formation cone, attractor basin, resonance lattice, graph, manifold, or analogous formal structure.	Moves from behavior and coupling to structure (geometry).
	7	Formal Metric / State Variable	Quantify the structure through variables such as coherence score, resonance index, formation stability, reification score, contextual coherence, or field-alignment metric.	Makes the theory measurable and testable.

C. IFII's Engineering Program and Preliminary Development Plan

Phase	Step	Element (What?)	Description (How?)	Linkage and Purpose (Why?)
	8	Constructive Dialectical Test	Challenge the primitive, coupling, geometry, and metric against rival explanations, reductionist models, phenomenological counterexamples, technical failure cases, and prototype stressors.	Prevents beautiful but false theory.
	9	Constructive Decision Rule	Accept, revise, downgrade, or reject the primitive / metric / geometry combination.	Creates a governance gate before axioms.
IV. The Pivot	10	Formation Axioms	State the validated principle as a formal axiom or axiom cluster that explains formation, stabilization, transformation, dissolution, or self-recognition.	Creates the handover from theory construction to engineering.
V. Deductive Engineering	11	Derived Claim	Deduce what must be true if the axiom holds.	Converts the axiom into propositional logic.
	12	H-D Hypothesis	Define falsifiable predictions about observable system behavior, prototype behavior, or experimental outputs.	Converts the claim into testable epistemology.
	13	Requirement (RSM ID)	Define formal shall statements, requirement subsets, acceptance criteria, and traceability IDs.	Converts theory into design obligations.
	14	Architecture Allocation / Design Pattern	Allocate requirements to software, hardware, data, interface, governance, training protocol, lab setup, or field instrumentation.	Bridges requirement and implementation.
	15	Systemic Dialectical Test	Stress-test the design under boundary conditions, counter-models, rival mechanisms, adversarial cases, and failure-mode scenarios.	Tests robustness and avoids self-confirming prototypes.
	16	Verification Artifacts	Produce proofs, simulations, tests, code, diagrams, test reports, experimental records, and review evidence.	Answers: Did we build the system right?
	17	Validation Protocol and Learning Loop	Validate against the original intent, user needs, field-informed criteria, and formation metrics; then feed results back to primitive status, axiom versioning, requirements, and design patterns.	Answers: Did we build the right system? It also refines both theory and artifact.

4. IFII's R&D Process

The construction grid details the fine-grained logical lineage of each claim about appearance, formation, referent-relation, or reality-status that IFII seeks to make traceable within the AFE and related research instruments. The R&D process defines the higher-level management phases, design-control elements, and governance gates used to execute that logic.

The CFM therefore functions as both a research method and a development process. It is a theory-construction method because it constructs and evaluates candidate principles of formation. It is an engineering process because it reformulates accepted candidate principles into requirements, architectures, prototypes, tests, and validation evidence. It is a governance process because every step is traceable, reviewable, and revisable.

The research and development work at IFII cannot rely solely on the standard hypothetico-deductive model, which assumes a set of abstract laws in advance and then deduces consequences. In this domain, the relevant candidate principles and formal representations are themselves under construction. IFII must first define, challenge, and test candidate models of formation before it can engineer systems that expose, evaluate, or use those models.

Consequently, IFII adopts constructive axiomatics as the front end of its development pipeline and systems-engineering discipline as the back end. Constructive axiomatics builds theoretical structures from carefully specified primitives rather than imposing them from above. Systems engineering then turns accepted axioms into claims, hypotheses, requirements, architectures, prototypes, and evidence.

The process does not ask only: "What are the irreducible observable primitives of this domain?" It asks: "Which candidate formation primitives recur across source corpus, lived experience, scientific analogy, computational analogy, failure modes, and prototype-exposable behavior, and how do they co-condition one another?"

This methodology is essential for IFII because the GTF is not a simple engineering specification and not a purely interpretive doctrine. It is a bounded reformulation research program. Its central challenge is to examine whether selected, source-reviewed structural insights can be reformulated into testable, shareable, ethically governable, and technically useful design constraints without implying spiritual equivalence or metaphysical proof.

C. IFII's Engineering Program and Preliminary Development Plan

4.1 The Constructive Front End: From Canon to Formation Axiom

The narrative of the R&D process begins not with a requirement, but with a search for the conventional dynamics, constraints, and traceable logic of formation. This is the domain of Loop 1: Constructive Validation.

Source Selection and Source-Corpus Building: The raw material for constructive axiomatics is the bounded source corpus and its review history. In standard engineering, inputs are user needs, market requirements, or known regulatory constraints. For the GTF, inputs are selected source materials, interpretations, phenomenological descriptions where appropriate, scientific analogues, and technical failure cases that may contain structurally relevant information about appearance, formation, context, and non-reification. IFII therefore requires a rigorous source-selection process to distinguish poetic metaphor, contemplative instruction, phenomenological description, technical doctrine, and potentially operationalizable principle. The result is a bounded, review-governed source corpus and supporting research corpus.

Candidate Primitive Constellation: From this source base, IFII identifies candidate primitives such as field, geometry, boundary condition, frequency, resonance, formation, memory, and recognition. These are treated as candidate research objects, not as final axioms. Each candidate must pass through the Seven-Lens Primitive Profile and Primitive Status Decision.

Primitive Status Decision: Each primitive is classified according to its evidential and operational status. It may be source-reviewed, phenomenological, scientific, computational, metaphorical, provisional, or rejected. This decision protects the project from overclaiming and makes explicit what kind of validity each primitive currently has.

Primitive Coupling Map: IFII then maps how primitives co-condition one another. This is the crucial formation move. A primitive does not operate alone. Boundary, frequency, resonance, formation, memory, and recognition may form a sequence, cycle, graph, field topology, or other relational structure. The coupling map expresses the interdependence that the GTF seeks to formalize.

Constructive Validation and Dialectical Testing: Once primitives and couplings are defined, they are challenged. IFII asks whether the proposed primitive or coupling is necessary, whether it can be reduced to a simpler explanation, whether rival models explain the same phenomenon better, and whether the proposed geometry survives phenomenological, scientific, computational, and failure-mode tests. This dialectical test is a formal attempt to break the theory before engineering begins.

Formation Axiom Handover: The output of Loop 1 is the pivot from theory construction to engineering. The formation axiom is a formal candidate statement that bridges source-informed theory construction and systems engineering. It acts as the handover contract between theoretical research and systems engineering. In the next phase, the axiom becomes an engineering constraint: not a loose inspiration, but a traceable design obligation.

Illustrative candidate axiom: "The apparent reality-status of a system-generated appearance or model object increases with the persistence, storage, repetition, and reinforcement of its formative pattern across a context-field." This is an example of axiom format, not a final GTF claim. In the CFM, a statement of this kind must be traceable to a primitive constellation, constructive tests, derived claims, hypotheses, requirements, and validation evidence.

4.2 Empirical Grounding, Failure Analysis, and Risk Management

Between the formation axiom handover and concrete architectural design lies a critical grounding layer: failure analysis and risk management. This ensures that engineering is not driven only by high theory. IFII does not design in a vacuum; it designs in response to specific, substantiated deficiencies in current systems and to the risks created by new formation-aware technologies.

IFII has identified a preliminary subset of empirical inputs about failure modes of current AI technologies for the design process. These include context drift, hallucination, brittle abstraction, reification, bias reinforcement, and loss of relational context. These failures are mapped against formation axioms to identify what the AFE or other research instruments must not fail at.

The design process therefore creates Failure-Derived Requirements (FDRs). Unlike ordinary requirements, which state what the system shall do, FDRs explicitly state what the system must not fail at. For example, if a candidate formation axiom implies that context is sustained by a relational field, while current architectures show context drift because they treat context primarily as a linear buffer, the resulting requirement may specify a context-state mechanism that remains stable under defined conditions.

Risk management is integrated at this stage. In IFII's context, risk includes technical failure, safety failure, governance failure, epistemic failure, and reality-status error. A system that generates unsupported appearances, confuses model-output with referent, inflates the reality-status of provisional formations, reinforces bias, or creates false validation evidence can corrupt the epistemic loop and lead to false confidence in the GTF. Risk controls must therefore be designed, implemented, verified, and included in validation protocols:

- **Hazard identification:** Identify hazards from failure modes, misuse scenarios, epistemic drift, and harmful system behavior.
- **Risk evaluation:** Assess severity, probability, detectability, and reversibility, including reality-status and epistemic severity.
- **Risk control measures:** Define mitigations in architecture, data, interface, governance, review workflow, and validation protocol.
- **Verification of effectiveness:** Confirm that risk controls work and that residual risk is acceptable for the intended research use.

4.3 The Deductive Back End: Engineering and Operationalization

Loop 2 begins once the formation axiom handover, failure analysis, and risk analysis provide traceable design inputs. This is the domain of systems engineering, design controls, architecture allocation, implementation, verification, validation, and learning return.

The model integrates four methodological movements:

1. Formation axioms and derived claims from Loop 1;
2. Failure-Derived Requirements from empirical grounding and research;
3. User needs and use-case constraints;
4. Risk control requirements;
5. Operational constraints such as measurement conditions, data boundaries, hardware limitations, governance rules, and validation metrics.

Requirements Engineering: These inputs are reformulated into engineering specifications in the Requirements Specification Matrix (RSM) and, where needed, a Design Requirements Matrix (DRM). Each requirement must be measurable, testable, traceable, and linked to an axiom, failure mode, user need, or risk control. Example requirement format: "The system shall preserve relational context across transformations under defined boundary conditions."

Architecture Allocation / Design Pattern: The model explicitly adds an architecture-allocation step after requirements. The key question is: Where does the requirement go? Software, hardware, interface, governance, data model, training protocol, experimental lab setup, or field instrumentation? Without this allocation, the process jumps too quickly from requirement to test and risks becoming unbuildable.

C. IFII's Engineering Program and Preliminary Development Plan

Requirement	Possible Architecture Allocation / Design Pattern
System shall preserve relational context across transformations.	Memory architecture, knowledge graph, context-state engine, or field-based memory substrate.
System shall identify reification events.	Reasoning monitor, uncertainty classifier, epistemic state tracker, or interpretability layer.
System shall expose formation dynamics visually.	Formation chamber UI, simulation layer, dynamic graph interface, or visual intelligence module.
System shall model resonance between prior state and current input.	State-space model, recurrent memory, attractor dynamics, resonance index, or signal-processing module.
System shall avoid self-confirming validation loops.	Independent test harness, rival-model benchmark, audit trail, review workflow, or dialectical stress-test protocol.

Design Implementation and Output: The design is then implemented as software, hardware, interface, simulation, lab setup, or another research instrument. Outputs may include source code, model configurations, schematics, architecture diagrams, datasets, prompts, protocols, test harnesses, and a Device-Master-Record-equivalent evidence package. The crucial distinction is traceability: IFII is not merely building; it is implementing requirements derived from formation axioms, failure analysis, user needs, and risk controls.

Systemic Verification: Verification asks, "Did we build the system right?" The activity is to test design outputs against design inputs. Examples include requirement compliance tests, simulations, component verification, baseline comparisons, risk-control tests, latency tests, coherence metrics, and failure-mode mitigation tests. If verification fails, the design loops back to implementation or requirements. A formation axiom is not changed merely because implementation failed; it is revised only if evidence shows that the axiom, claim, hypothesis, or metric is invalid or under-specified.

Systemic Validation and Learning Return: Validation asks, "Did we build the right system?" Because IFII is a generational research program, validation is iterative and cumulative. Validation tests whether the prototype satisfies the original intent, addresses the relevant failure modes, produces coherent effects under real or simulated use conditions, and provides evidence relevant to the GTF. The result feeds back into the primitive registry, axiom set, requirements baseline, architecture patterns, risk controls, and research agenda.

C. IFII's Engineering Program and Preliminary Development Plan

4.4 The Integrated R&D Process Table

The following table translates the detailed construction grid into IFII's high-level R&D management process and governance gates:

Step	Phase	Design Control Element	Activity / Process	Input	Output / Deliverable	Risk / Quality Gate
1	Orientation and Source Boundary	Design Planning	Define domain, research object, source corpus, scientific corpus, use-case context, and intended research instrument.	Strategic mandate, mission, texts, theories, terminologies, research scope.	Domain definition, bounded, review-governed source corpus, scientific corpus, RAG / database scope, use-case boundary.	Scope gate: Prevent overgeneralization and ensure sources are suitable for responsible research reformulation.
2	Primitive Discovery and Registry	Research / Concept	Identify candidate primitive constellation and create Seven-Lens Primitive Profiles.	Bounded source corpus, phenomenological material, scientific analogues, computational analogues, failure-mode data.	Primitive registry entries, primitive status records, traceability links.	Status gate: Classify as source-reviewed, phenomenological, scientific, computational, metaphorical, provisional, or rejected.
3	Constructive Formation and Coupling	Research / Concept	Define operational behavior, map primitive couplings, derive geometric or topological structures, and define metrics or state variables.	Primitive profiles and status decisions.	Primitive coupling map, geometric schematics, metric definitions, candidate state variables.	Constructive test: Challenge against rival models, counterexamples, reductionist explanations, and technical failure cases.

C. IFII's Engineering Program and Preliminary Development Plan

Step	Phase	Design Control Element	Activity / Process	Input	Output / Deliverable	Risk / Quality Gate
4	Dialectical Axiom Gate / Formation Axiom Handover	Design Input / Theory Transfer	Formalize accepted primitive/metric structures into formation axioms and transfer them to engineering as design constraints.	Validated primitive constellation, coupling map, metric, constructive decision rule.	Formation axiom set, axiom version record, handover contract, initial derived claims.	Axiom gate: Axiom must be clear, traceable, falsifiable through derived hypotheses, and usable as an engineering constraint.
5	Failure Analysis and Risk Management	Risk Analysis	Analyze current-paradigm failure modes; map axioms to failures; define FDRs and risk controls.	Research, benchmarks, incident patterns, literature, prototype limitations.	Failure-mode taxonomy, Failure-Derived Requirements, risk management plan, risk-control requirements.	Risk gate: Prioritize by severity, including technical, safety, epistemic, and ontological risk.
6	Requirements and Architecture Allocation	Design Input / Architecture	Translate axioms, FDRs, user needs, and risk controls into measurable requirements; allocate each requirement to architecture or design pattern.	Formation axioms, FDRs, user needs, risk controls, operational constraints.	Requirements Specification Matrix, Design Requirements Matrix, architecture allocation records, design patterns.	Requirements review: Ensure requirements are unambiguous, measurable, testable, and allocated.
7	Design Implementation	Design Output	Build or configure software, hardware, interface, simulation, lab setup, or research instrument.	RSM, DRM, architecture allocation, design patterns, protocols.	AFE prototype or related instrument, source code, schematics, datasets, prompts,	Design review: Confirm design outputs trace back to design inputs and risk controls.

C. IFII's Engineering Program and Preliminary Development Plan

Step	Phase	Design Control Element	Activity / Process	Input	Output / Deliverable	Risk / Quality Gate
					test harnesses, evidence package.	
8	Systemic Verification	Design Verification	Test the implemented design against requirements, specifications, architecture allocations, and risk controls.	Prototype, test protocols, requirements, acceptance criteria.	Verification artifacts, test reports, simulation outputs, review records.	Verification gate: Acceptance criteria must be met or deviations must trigger corrective loops.
9	Systemic Validation and Learning Return	Design Validation / Knowledge Governance	Validate the prototype under intended use or stress conditions; assess whether it satisfies the original intent and informs the GTF.	Verified prototype, validation protocols, field-informed metrics, user/use-case evidence.	Validation report, operational metrics, learning-return record, updated primitive registry, axiom version, RSM, and architecture patterns.	Epistemic gate: Determine whether evidence corroborates, weakens, falsifies, or refines the relevant formation axioms and requirements.

5. CFM Preliminary Design and Development Plan

The Constructive Formation Model software should not primarily be a chatbot or a generic RAG database. Ultimately, it should be a traceability and governance system for theory construction. Its core function is to preserve the chain of custody from bounded source corpus to primitive identification, primitive profiling, status decision, coupling map, axiom construction, derived claim, hypothesis, requirement, architecture allocation, verification evidence, validation result, and learning return.

THE CFM SW TOOL SHOULD FUNCTION AS A HYBRID OF:

Source-corpus repository, research knowledge base, primitive registry, axiom construction workbench, requirements-management system, V&V evidence system, change/version-control system, review-gate workflow.

ASSUMPTIONS:

The CFM software tool MVP will be developed over a 33-month project period as a bounded research infrastructure prototype, not as a finished enterprise-grade platform or complete AFE implementation. The project assumes that the first 24 months will focus on building the core CFM software workbench, while months 25-33 will apply the tool to one paradigmatic prototype case that demonstrates the source-corpus-to-primitive-to-axiom-to-requirement-to-validation logic.

The MVP will support one bounded source-corpus subset, one initial primitive constellation, seven-lens primitive profiling, primitive status decisions, primitive coupling maps, candidate formation axioms, derived claims, H-D hypotheses, requirements with RSM IDs, architecture allocations, validation protocols, learning-return records, and change/version control. The goal is to demonstrate a transparent, testable, and reusable research process, not to finalize the General Theory of Formation or automate philosophical judgment.

The project assumes a hybrid human-AI workflow. AI assistance may be used for source indexing, semantic search, candidate extraction, clustering, comparison, drafting, contradiction detection, and traceability support. However, all source interpretation, primitive status decisions, axiom acceptance, requirements baselines, and validation conclusions remain subject to human review and approval. The tool is therefore AI-assisted, human-governed, and source-fidelity-protected.

The project further assumes that collaborators will contribute at different intensities and for different durations. Some roles may be continuous but part-time, such as the Project Leader, Technical Lead, and V&V Lead. Other roles may be periodic or milestone-based, such as source-fidelity reviewers, source-corpus SMEs, science

SMEs, computational SMEs, UX support, and prototype reviewers. The development plan must therefore rely on clear role definitions, review gates, asynchronous documentation, and controlled handover artifacts rather than continuous full-time availability from all contributors.

The project assumes that IFII will prioritize a small number of high-value roles early: Project Leader / CFM Product Owner, Technical Lead, Knowledge Engineer or RAG Developer, Requirements/V&V Lead, source-fidelity reviewer, and Prototype Lead. Additional specialists may be onboarded for defined work packages once the data model, source boundary, and MVP workflow are stable.

CONSTRAINTS:

The main project constraint is scope control. The MVP cannot cover the full source corpus, all possible primitive constellations, all scientific analogues, all prototype domains, or a mature AFE implementation. The first software prototype must remain deliberately narrow: one bounded source corpus, one initial primitive constellation, one axiom-construction pathway, one requirements subset, and one paradigmatic prototype case.

The second constraint is collaborator availability. Because key contributors may work part-time, intermittently, or only for defined periods, the project cannot depend on continuous synchronous collaboration. The development process must include structured documentation, modular work packages, decision logs, review records, versioned baselines, and clear handover points. Every major artifact must be understandable to later contributors without relying on undocumented oral context.

The third constraint is epistemic authority. The software may assist with source-corpus processing and candidate generation, but it must not present generated outputs as authoritative. Source-fidelity review, primitive classification, axiom acceptance, and contemplative interpretation require qualified human review. The system must distinguish clearly between source-reviewed evidence, phenomenological interpretation, scientific analogy, computational analogy, metaphorical material, provisional inference, and technical requirement.

The fourth constraint is technical maturity. The project should not promise fully automated theory construction, autonomous axiom generation, mature graph reasoning, enterprise-grade requirements management, or production-ready AI agent architecture within the 33-month period. The realistic target is an auditable research workbench with traceability, version control, review workflows, and enough AI assistance to make the process scalable and inspectable.

The fifth constraint is validation maturity. During the project period, validation will be early-stage and paradigmatic. The project can test whether the CFM process produces coherent, traceable, challengeable, and prototype-exposable design intelligence. It cannot yet prove the full General Theory of Formation, establish final formation axioms, or validate field-informed intelligence across broad technological domains.

The sixth constraint is source rights and data governance. Source materials, reformulations, commentaries, notes, and research corpora may have copyright, lineage, access, or ethical-use restrictions. The software architecture must therefore support source metadata, access control, citation, provenance, and usage restrictions from the beginning. Public release should focus on open documentation, public-domain or permissioned materials, schemas, methods notes, and selected non-restricted outputs.

The seventh constraint is change and version control. Because the project's core objects – sources, primitives, axioms, requirements, and validation protocols – may evolve through review and learning return, uncontrolled revision would undermine the project's credibility. Versioning, change requests, review gates, baseline control, and impact analysis are therefore not optional features; they are core requirements of the MVP.

The eighth constraint is prototype selection. The final prototype phase must be narrow enough to be achievable within nine months. The paradigmatic prototype should therefore demonstrate one clearly bounded use case, such as a context-coherence monitor, reification-detection module, formation-dynamics visualizer, or dialectical stress-test harness. It should demonstrate the CFM pipeline, not attempt to instantiate the full AFE vision.

PLANNING IMPLICATION:

Given these assumptions and constraints, the project should be managed as a staged MVP: first build the governed knowledge and traceability backbone, then the primitive registry, then the axiom-construction and requirements bridge, and only then apply the tool to one paradigmatic prototype. Success should be defined as a working, reviewable, and extensible research infrastructure that demonstrates the CFM logic end-to-end, rather than as a finished theory, complete platform, or fully validated field-informed intelligence system.

C. IFII's Engineering Program and Preliminary Development Plan

5.1 Time Planning

IFII plans to develop the automated CFM pipeline MVP over a period of 33 months. The realistic goal is a traceable *source-corpus → primitive → axiom → requirement → prototype-evidence* workbench. The first two years are treated as an MVP / research infrastructure build, not as a finished enterprise-grade platform:

- **Years 1-2:** Build the CFM tool as a controlled, traceable research workbench for one bounded domain and one primitive constellation.
- **Year 3:** Use that tool to build and validate one paradigmatic prototype

SCOPE:

The prototype will support one bounded source-corpus subset, one initial primitive constellation, seven-lens primitive profiling, primitive status decisions, coupling maps, candidate formation axioms, derived claims, H-D hypotheses, requirements with RSM IDs, architecture allocations, validation protocols, learning-return records, and change/version control.

SUCCESS CRITERIA:

- IFII chooses one bounded source-corpus subset.
- IFII starts with 5-8 primitives, not 30.
- The software begins as a workflow plus traceability plus registry system, not an all-knowing AI system.
- AI assistance is used for extraction, summarization, candidate linking, and drafting - but not for final authority (human oversight).
- Change/version control is built from day one.
- There is one stable Technical Lead across most of the 24 months.
- Source-fidelity reviewers and SMEs work through scheduled review gates, not continuous involvement.
- The Year 3 prototype is narrow and paradigmatic.

FAILURE CRITERIA:

- Source-corpus boundary and review workflow remain undefined after month 6.
- No one owns the data model and architecture.
- Every collaborator only contributes episodically.

- The tool tries to automate philosophical judgment too early.
- Requirements engineering is postponed until year 3.
- The paradigmatic prototype is too ambitious.
- "AFE prototype" is interpreted as a full system rather than a demonstrator.

TIME PLAN:

Year 1 – Build the CFM Research Backbone			
Phase	**Time**	**Main Build**	**Output**
0	Months 0-2	Scope, data model, role model, architecture, MVP definition	CFM software specification and object model
1	Months 2-6	Source-corpus repository, source-review workflow, basic versioning	Source-corpus MVP
2	Months 4-9	Primitive registry, seven-lens profiles, status decisions	Primitive registry MVP
3	Months 7-12	Primitive coupling maps, operational behavior, metric/state-variable records	Constructive Formation workspace v1

Year 2 – Build the Theory-to-Engineering Bridge			
Phase	**Time**	**Main Build**	**Output**
4	Months 10-15	Axiom construction, derived claims, H-D hypotheses, axiom handover	Formation axiom module
5	Months 13-18	Failure-mode taxonomy, risk register, Failure-Derived Requirements	Risk/FDR module
6	Months 16-21	Requirements Specification Matrix, architecture allocation, baseline control	RSM module
7	Months 19-24	Verification/validation evidence, learning return, reporting/export	End-to-end CFM prototype

Target: By month 24, the realistic target is a working end-to-end CFM prototype that can process one bounded source-corpus subset and one primitive constellation into candidate axioms, derived claims, requirements, review gates, and prototype-ready design inputs. It should not be expected to handle every source text, every primitive, every scientific analogy, and every possible prototype domain.

Year 3 – One Paradigmatic Prototype	
Prototype Option	**Suitability**
Context-coherence AI agent	Tests context drift, relational memory, reification detection
Reification monitor	Detects when an AI system turns relational/process information into fixed object claims, collapses appearance into referent, or inflates a provisional model object into final reality-status.
Formation dynamics visualizer	Shows field, boundary, resonance, appearance, referent/evidence, model object, actualized formation, memory, recognition, and release as a visual model
Field-informed memory substrate prototype	Tests whether context can be modeled non-linearly rather than as a linear buffer
Dialectical stress-test harness	Tests whether axioms/requirements avoid self-confirming validation loops

Target: Year 3 should produce one prototype, one requirements baseline, one verification protocol, one validation protocol, one evidence package, one learning-return report, one public methods note.

5.2 Project Team

CORE TEAM:

Role	Needed When	Main Responsibility
Project Leader / CFM Product Owner	Full 3 years, part-time possible but consistent	Owns vision, priorities, review gates, funder reporting, conceptual integrity
Technical Lead / Software Architect	Months 0-24, ideally 40-60%	Owns system architecture, data model, integrations, code quality
Full-Stack Developer	Months 2-24, 50-100% depending scope	Builds application, workflows, UI, database, export functions
Knowledge Engineer / RAG Developer	Months 2-18, 30-60%	Source ingestion, metadata, semantic search, traceability, AI-assisted extraction
Requirements / V&V Lead	Months 6-36, 20-50%	RSM, verification, validation, risk controls, evidence package
Prototype Lead	Months 18-36, 40-80%	Builds the paradigmatic prototype and test harness

C. IFII's Engineering Program and Preliminary Development Plan

Role	Needed When	Main Responsibility
UX / Product Designer	Months 0-6 and 12-18, short bursts	Makes complex workflows usable
DevOps / Data Security Support	Periodic	Repository, access, backups, deployment, permissions

EXPERT REVIEWER:

Role	Needed When	Main Responsibility
Source-Fidelity Reviewer	Intermittent over 3 years	Final source-fidelity review and guard against misuse, overreach, or misinterpretation
Source-Corpus SME	Months 2-18, recurring	Source selection, primitive interpretation, source comments, and boundary review
Science SME	Months 6-24, periodic	Scientific analogues, limits of analogy, metric plausibility
Computational / AI SME	Months 6-36, periodic	Computational analogues, prototype architecture, AI failure modes
Ethics / AI Safety SME	Months 12-36, periodic	Risk controls, misuse, governance, validation safeguards

5.3 Design and Development Planning

5.3.1 CFM Backbone: Data Model, Roles, Workflow, Versioning

Core Object	Purpose
Source	Source text, reformulation, commentary, paper, interview, image, audio, or other evidence item admitted into the bounded source corpus for a specific research cycle
Source Excerpt	Specific passage or unit of source material
Candidate Primitive	Example: field, boundary condition, resonance, formation, memory, recognition
Seven-Lens Profile	Source-corpus, phenomenological, scientific, computational, failure-mode, requirement-seed, prototype-exposure profile
Primitive Status Decision	Source-reviewed, phenomenological, scientific, computational, metaphorical, provisional, rejected
Primitive Coupling	Relation between primitives

C. IFII's Engineering Program and Preliminary Development Plan

Core Object	Purpose
Metric / State Variable	Coherence score, resonance index, reification score, etc.
Formation Axiom	Candidate principle derived from primitive constellation
Derived Claim	What must be true if the axiom holds
H-D Hypothesis	Testable prediction
Requirement	Formal "shall" statement with RSM ID
Failure-Derived Requirement	Requirement derived from current technology failure modes
Appearance / Presencing Record	Captures what appears to the user or system: output, interface state, visualization, recommendation, interpretation
Appearing Referent / Evidence Object	Captures what the appearance claims to be about: source passage, observation, sensor datum, external constraint, user context, system state
Mental / Model Object	Captures the conceptual or computational object constructed from source, evidence, interpretation, and assumptions
Actualized Formation	Captures the concrete runtime manifestation: displayed answer, prototype event, recommendation, decision support state, dashboard object
Reality-Status Marker	Marks whether the object is provisional, conditioned, contested, validated, deprecated, revoked, or released
Risk Control	Governance, design, data, architecture, or validation control
Architecture Allocation	Where the requirement is implemented
Verification Artifact	Test, simulation, proof, code, diagram, review record
Validation Protocol	Method to assess whether the system satisfies original intent
Learning Return	Evidence-based update to primitive, axiom, requirement, or design pattern
Change Request	Controlled proposal to revise a source boundary, primitive, axiom, requirement, or artifact

PROJECT TEAM:

CFM Product Owner / Project Leader, Technical Lead, Data Architect, Requirements / V&V Lead, UX designer.

5.3.2 Canon-Building Module (Source-Governance Layer)

Function	Action
Source ingestion	Upload/import texts, reformulations, commentaries, articles, notes
Metadata management	Author, tradition, lineage, language, reformulation status, reliability, rights
Source boundary definition	Defines what is inside / outside the authorized corpus
Source-corpus review workflow	Draft → reviewed → approved → excluded
Passage-level citation	Allows each primitive to trace back to exact passages
Source-fidelity comments	Source-fidelity reviewer / source-corpus SME can approve, qualify, or reject interpretations
RAG index (later)	Useful, but only after metadata and source boundaries are stable

PROJECT TEAM:

Knowledge Engineer, Research Librarian / Source-Corpus Curator, Source-Fidelity Reviewer, Source-Corpus SME, NLP/RAG Developer, Full-Stack Developer.

MINIMUM VIABLE VERSION:

A structured repository with approved source excerpts, metadata, status, and traceability links. Full AI-assisted source-corpus analysis can come later.

5.3.3 Primitive Registry Module

Function	Action
Operational behavior definition	Defines what the primitive does: opens, constrains, resonates, stabilizes, remembers, recognizes, dissolves
Coupling map	Visual or tabular map of how primitives co-condition each other
Geometry/topology notes	Formation cone, attractor basin, graph, field topology, lattice, mandalic structure, etc.
Metric/state-variable registry	Defines measurable indicators such as coherence, resonance, reification, contextual stability
Constructive dialectical test	Records rival explanations, counterexamples, reductionist alternatives, failure cases

Function	Action
Constructive decision rule	Accept, revise, downgrade, or reject a primitive/metric/geometry combination

PROJECT TEAM:

CFM Theorist, Systems Scientist / Mathematician, AI/Computational SME, Knowledge Engineer, UX Designer, Full-Stack Developer.

MINIMUM VIABLE VERSION:

A structured relational table plus simple network visualization.

5.3.4 Axiom Construction and Formation Axiom Handover Module

Function	Action
Axiom drafting template	Converts primitive constellation into formal axiom candidate
Backward traceability	Shows source → primitive → profile → coupling → metric → axiom
Derived claim generation	What follows if the axiom holds
H-D hypothesis creation	Testable predictions
Axiom review gate	source-fidelity review, scientific review, engineering review
Axiom versioning	Axiom v0.1, v0.2, accepted/revised/rejected
Axiom handover package	Exportable handover to requirements engineering

PROJECT TEAM:

CFM Product Owner, Source-Corpus SME, Scientific SME, Systems Engineer, V&V Lead, Full-Stack Developer.

MINIMUM VIABLE VERSION:

Structured axiom records with traceability and review decisions. AI-assisted drafting can be included, but human review must remain the gate.

5.3.5 Failure Analysis and Risk-Management Module

Function	Action
Failure-mode taxonomy	Stores known AI/system failure modes
Failure-to-axiom mapping	Connects failure modes to relevant formation axioms
Failure-Derived Requirements	Defines what the system must not fail at
Risk register	Technical, safety, governance, epistemic, ontological risks
Risk evaluation	Severity, probability, detectability, reversibility
Risk controls	Architecture, data, interface, governance, review, validation controls

PROJECT TEAM:

V&V Lead, AI Safety / Risk SME, Systems Engineer, Quality/Regulatory Mindset Lead, Technical Lead.

MINIMUM VIABLE VERSION:

Risk register, failure-mode taxonomy, FDR generation linked to axioms.

5.3.6 Requirements Specification Matrix Module

Function	Action
Requirement creation	Formal "shall" statements
RSM ID generation	Unique requirement IDs
Acceptance criteria	Defines how each requirement is tested
Traceability	Axiom / claim / hypothesis / failure mode / user need / risk control → requirement
Architecture allocation	Software, data, interface, governance, lab setup, simulation, hardware
Requirements review	Clear, measurable, testable, allocated
Baseline management	Freezes a set of requirements for prototype work

PROJECT TEAM:

Systems Engineer, Requirements Engineer, V&V Lead, Software Architect, Prototype Lead.

MINIMUM VIABLE VERSION:

A robust requirements table with IDs, links, acceptance criteria, allocation, review status, and version history.

5.3.7 Verification, Validation, and Learning-Return Module

Function	Action
Test protocol creation	Defines verification tests
Verification artifact upload	Code, simulation, test result, diagram, review record
Validation protocol creation	Tests original intent and formation metrics
Evidence review	Accepted / insufficient / contradictory / falsifying
Learning return	Pushes findings back into primitives, axioms, requirements
Corrective loop	Triggers change request or revision
Export package	Evidence bundle for funders, reviewers, collaborators

PROJECT TEAM:

V&V Lead, Prototype Lead, Software Developers, Test Engineer, CFM Product Owner, Scientific and Source-Fidelity Reviewers.

MINIMUM VIABLE VERSION:

Evidence records, review outcomes, and learning-return links.

5.3.8 Change and Version-Control System Element

Function	Action
Version history	Sources, primitive profiles, axioms, requirements, protocols
Change request workflow	Proposed change → impact analysis → review → approve/reject

C. IFII's Engineering Program and Preliminary Development Plan

Function	Action
Baselines	Source-corpus baseline, primitive baseline, axiom baseline, RSM baseline
Impact analysis	What changes if this primitive/axiom/requirement changes?
Audit trail	Who changed what, when, why
Git integration	For code, schemas, protocols, public repository
Exportable review package	Useful for reporting, academic review, and public documentation

PROJECT TEAM:

Technical Lead, Quality/V&V Lead, Software Architect, Full-Stack Developer.

MINIMUM VIABLE VERSION:

Versioned records, change requests, review decisions, and traceability impact view.

6. Applying the CFM Derived Requirements Subset in the AFE

The AFE applies the CFM-derived requirements subset as a bounded design input baseline. The subset should not be treated as a complete AFE specification or as final proof of the GTF. It is a first engineering handover package: a small, traceable set of requirements derived from selected formation axioms, failure modes, risk controls, and prototype-exposure criteria. Its purpose is to make one narrow AFE vertical slice buildable, testable, reviewable, and correctable within the project period.

The requirements subset functions as a contract between the constructive front end and the deductive engineering back end. Upstream, each requirement must link back to source-corpus evidence, primitive profiles, axiom records, failure modes, risk controls, and review decisions. Downstream, each requirement must link forward to architecture allocation, implementation element, verification method, validation protocol, and learning-return record.

6.1 Requirement Subset as AFE Design Input

The first AFE requirement subset should focus on formation accountability rather than on agentic autonomy. A minimal subset can be organized around eight requirement families. Each family expresses a formation principle, an implementation obligation, and a verification/validation pathway

C. IFII's Engineering Program and Preliminary Development Plan

Requirement Family	CFM / GTF Source Logic	AFE Capability	Evidence and Validation Purpose
Formation traceability	No system-generated appearance or actualized formation should be presented without traceable causes, conditions, referents, assumptions, and boundary conditions.	Every significant system-generated appearance, model object, recommendation, interface state, or actualized formation links to sources, referents/evidence, assumptions, primitives, axioms, requirements, context, and boundary conditions.	Traceability matrix, formation graph, source map. Validates reduction of black-box opacity.
Provisionality / non-reification	Unrecognized formations tend toward reification.	Outputs are explicitly marked as provisional, conditioned, incomplete, and contestable.	UI inspection, user interpretation tests, reification score. Validates non-final status of outputs.
Context integrity	A formation is sustained by its field and boundary conditions.	The system preserves source context, user context, institutional context, and consequence field.	Context-coherence test, boundary-condition record. Validates resistance to context drift.
Dialectical contestability	Constructive and systemic dialectical tests prevent self-confirming theory.	The system exposes counterviews, rival explanations, failure modes, and uncertainty.	Counterview logs, stress-test records. Validates openness to correction.
Release / dissolution	A healthy formation can transform or dissolve when conditions change.	Outputs, models, profiles, and recommendations have expiration, revision, revocation, forgetting, or archival pathways.	Decay logs, rollback tests, revision records. Validates non-capture and reversibility.
Responsiveness / compassion	Beneficial intelligence remains responsive to relation, agency, harm, and whole-system effects.	Impact checks assess agency, dependency, harm, relational coherence, and downstream consequences.	Agency rubric, harm review, impact accountability view. Validates responsible

Requirement Family	CFM / GTF Source Logic	AFE Capability	Evidence and Validation Purpose
			participation, not mere optimization.
Source fidelity and human governance	Source-derived concepts require qualified human review.	source-fidelity review, scientific review, engineering review, and V&V review gates control acceptance.	Review records, decision logs, version history. Validates non-appropriative reformulation.
Capability containment	Formation insight is dual-use if stripped of ethics and governance.	High-risk design patterns, manipulative uses, spiritual-authority claims, and unrestricted release pathways are blocked or controlled.	Misuse taxonomy, access controls, prohibited-use checks. Validates safety doctrine.

6.2 The AFE Reference Core and Formation Graph

The first AFE should be a reference core, not a public product or autonomous agent. The reference core is the minimal research architecture that can perform formation-accountability operations under explicit boundary conditions. It should include a formation graph, source and evidence links, requirement links, risk-control links, interface disclosure mechanisms, validation evidence, and learning-return records.

This turns source-informed questions about dependent arising and misrecognition into a conventional traceability architecture without pretending to model or exhaust their source meaning. The formation graph does not claim to describe reality-as-such. It records how a specific system-generated formation arose under specified conditions, what it refers to, what assumptions it uses, what evidence supports it, what uncertainty remains, and how it can be revised or released.

The reference core can be described as a layered architecture. These layers are not yet a finished AFE system. They define the minimal technical, governance, and interface functions required to test formation-accountability behavior.

Reference-Core Layer	Core Function	Requirement Families Implemented
Source layer	Stores source passages, interpretations, reviewer status, permissions, and source boundary decisions.	Source fidelity, traceability, human governance

C. IFII's Engineering Program and Preliminary Development Plan

Reference-Core Layer	Core Function	Requirement Families Implemented
Primitive layer	Holds candidate primitives, seven-lens profiles, status decisions, and coupling maps.	Traceability, context integrity, category-error prevention
Axiom layer	Stores candidate formation axioms, derived claims, H-D hypotheses, and axiom version records.	Contestability, review gates, learning return
Requirements layer	Stores RSM IDs, acceptance criteria, architecture allocation, risk controls, and baselines.	Design control, verification readiness, change control
Formation graph layer	Links source → appearing referent/evidence → appearance/model object → primitive → axiom → requirement → design pattern → actualized formation/runtime event → impact → revision/release.	Formation traceability, context integrity, learning return
Context field model	Represents actors, constraints, history, environment, use-case boundary, and downstream consequences.	Context integrity, responsiveness, impact assessment
Recognition layer	Marks system-generated appearances, model objects, and actualized formations as formed, provisional, conditioned, incomplete, referent-limited, and contestable.	Non-reification, uncertainty visibility, moral agency preservation
Release layer	Enables expiration, revision, forgetting, rollback, revocation, and archival as historical trace.	Release/dissolution, anti-capture, reversibility
Responsiveness layer	Checks harm, dependency, agency, relational coherence, institutional effects, and whole-system consequences.	Compassion/responsiveness, safety, impact validation
Accountability-view layer	Makes source, condition, assumption, counterview, impact, and release visible to human users.	Visibility, user agency, formation accountability
V&V layer	Runs verification tests, validation protocols, stress tests, evidence review, and learning-return decisions.	Verification, validation, dialectical challenge, quality governance

6.3 The Lamp Interfaces as User-Facing Formation Accountability

The AFE's interface should not hide its formation logic behind a fluent answer. It should make the relevant formation conditions visible through accountability views. In internal shorthand, these may still be called "lamp interfaces," but the external meaning is technical: interface functions that illuminate source, condition, assumption, uncertainty, counterview, impact, and release. The term does not imply spiritual vision, access to ultimate truth, or machine realization.

Accountability View (Lamp)	Question Answered for the User	Engineering Expression Ideas
Source lamp	Where does this output come from?	Source map, citation chain, source status, reviewer status
Condition lamp	Which context and boundary conditions shaped it?	Context card, scope statement, boundary-condition record
Assumption lamp	What must be true for it to hold?	Assumption list, uncertainty statement, confidence limits
Counterview lamp	What else could be true?	Rival explanations, failure modes, dialectical challenges
Impact lamp	What could this output do in the world?	Agency, harm, dependency, relational and institutional effects
Referent / Evidence Lamp	What does this appearance claim to be about?	Referent map, evidence object, source/evidence distinction, sensor/source/context link
Reality-Status Lamp	What is the status and limit of this appearance or formation?	Provisionality marker, validation state, uncertainty, deprecated/revoked status, "not reality-as-such" guardrail
Release lamp	How can this formation be changed or dissolved?	Expiration, revision, rollback, forgetting, archival, change request

6.4 Paradigmatic Vertical Slice: Formation-and-Release Interface

A suitable first prototype is a formation-and-release Interface. Its purpose is to demonstrate that a system-generated appearance or actualized formation can arise with visible causes, referents/evidence, assumptions, boundary conditions, and reality-status limits, serve a bounded function, remain provisional and contestable, transform under new information, and dissolve when no longer valid. This prototype is intentionally modest. It does not attempt to build the full AFE. It tests whether the CFM-

derived requirements subset can be implemented in one coherent user-facing workflow.

Formation Lifecycle Step	Prototype Behavior	Verification Evidence
Arising	The system-generated appearance arises together with source, context, constraints, assumptions, and requirement links.	Formation graph instance; traceability matrix entry
Stabilization	The user may temporarily stabilize or use the actualized formation, but must state purpose and scope.	Pinning rationale; use-case boundary record
Recognition	The interface marks the output as conditioned, incomplete, and challengeable.	Recognition marker; uncertainty and counterview inspection
Transformation	The appearance/model object can be transformed through another primitive, new source, counterview, or changed condition.	Revision log; alternative formation graph branch
Dissolution	The output expires, is revoked, forgotten, or archived as a historical trace rather than treated as an active or authoritative formation.	Expiration/rollback/archival log; release test result

6.5 Verification, Validation, and Learning Return

Verification asks whether the AFE reference core implements the specified requirements. Validation asks whether the implemented system serves the formation-aware purpose for which the requirements were derived. Learning return asks what the evidence means for the primitive registry, axiom version, requirements baseline, validation metrics, and future prototype architecture.

Question Type	Guiding Question	Indicative Evidence
Verification	Did we build the system right?	Requirement test results, traceability matrix, release-layer tests, human-review records, access-control checks
Validation	Did we build the right system for the intended formation-aware purpose?	User interpretation tests, context-coherence measures, agency rubric, dark-pattern audit, expert review

Question Type	Guiding Question	Indicative Evidence
Dialectical stress test	Can the theory or artifact be broken by rival explanations, misuse scenarios, or failure modes?	Counter-model review, source-fidelity review, AI-safety critique, psychological-safety review, institutional-misuse analysis
Learning return	What must change in the GTF, CFM, requirements, or AFE architecture?	Updated primitive status, axiom version, RSM revision, corrective action, public methods note

7. Preliminary AFE Engineering Roadmap

The AFE should not be built first as an "AI system." It should be built as a formation research instrument: a system-of-systems that lets IFII test whether system-generated appearances, mental/model objects, actualized formations, decisions, interface states, and actions can be generated, traced to referents/evidence and conditions, challenged, released, and revised without becoming reified capture-objects. The roadmap should therefore move from view to primitives, from primitives to axioms, from axioms to requirements, from requirements to architecture, from architecture to prototype, from prototype to validation, and from validation back into learning return.

The 33-month roadmap is designed as a bounded proof of method: long enough to demonstrate the CFM-to-AFE reformulation chain, but deliberately limited so that IFII does not overclaim completion of the GTF, a full AFE, or a general field-informed intelligence system.

For a preliminary 33-month project frame, the realistic target is not a complete AFE, not a public consumer product, and not an autonomous field-informed intelligence. The target is one AFE-relevant vertical slice: a bounded prototype that demonstrates how the CFM tool can reformulate one source subset and one primitive constellation into candidate axioms, requirements, an AFE reference core design, verification evidence, validation evidence, and a learning-return package.

7.1 Roadmap Principles and Phase Logic

Roadmap Principle	Engineering Implication
Start with boundary, not capability	Define what AFE is and is not before building features. Exclude sentience claims, guru-system behavior, consumer deployment, and metaphysical proof claims.

Roadmap Principle	Engineering Implication
Build traceability before intelligence	The first reference core must show how system-generated appearances and actualized formations arise from sources, referents/evidence, assumptions, context, constraints, requirements, and design decisions; it should not optimize fluent output before traceability, review, and release are in place.
Prototype one axiom at a time	Vertical slices should test one formation axiom or requirement family rather than attempting broad AFE functionality.
Design for release from day one	Expiration, rollback, revision, forgetting, and archival must be core architecture, not later ethical add-ons.
Keep human review authoritative	AI assistance may support extraction, linking, drafting, clustering, and contradiction detection, but not source meaning, axiom acceptance, or validation conclusions.
Make validation adversarial	Each prototype must be tested against source-fidelity concerns, scientific critique, engineering failure, AI-safety concerns, psychological safety, and institutional misuse.
Treat the AFE as public-good infrastructure	The output should be standards, reference architecture, methods notes, evidence packages, and safe open-core documentation, not a closed persuasion engine.

7.2 33-Month Bounded Proof of Method

The AFE roadmap should align with the existing CFM development plan: months 0-24 build the governed CFM workbench and theory-to-requirement bridge; months 25-33 use it to produce one paradigmatic AFE-relevant prototype case. AFE development during this period should therefore be deliberately dependent on CFM maturity.

Timeframe	AFE-Facing Objective	Primary Deliverables	Decision Gate
Months 0-3	Define AFE boundary and safety doctrine.	AFE charter; non-sentience declaration; prohibited-use list; source/bridge/analogy/implementation language policy; initial risk taxonomy.	Scope gate: AFE is confirmed as non-sentient research instrument, not a guru-system or consumer app.
Months 3-6	Prepare AFE-relevant source and primitive inputs.	Bounded source subset; initial primitive constellation; source-	Source gate: terms are classified before engineering reformulation begins.

Timeframe	AFE-Facing Objective	Primary Deliverables	Decision Gate
		fidelity workflow; category-error register.	
Months 6-12	Develop primitive profiles and candidate formation axioms relevant to one prototype domain.	Seven-lens profiles; coupling maps; operational behaviors; 3-5 candidate formation axioms; derived claims.	Axiom gate: only traceable, reviewable, challengeable axioms proceed.
Months 12-18	Translate axioms and failure modes into the first AFE requirements subset.	RSM entries; Failure-Derived Requirements; risk controls; acceptance criteria; architecture-allocation candidates.	Requirements gate: each requirement is measurable, testable, allocated, and traceable.
Months 18-24	Specify the AFE reference core and vertical-slice prototype architecture.	Formation graph design; context field model; lamp interface design; release layer; V&V protocol; evidence model.	Prototype-readiness gate: the slice is narrow enough for months 25-33.
Months 25-30	Implement one paradigmatic vertical slice.	Formation-and-release interface, reification monitor, context-coherence module, formation-dynamics visualizer, or dialectical stress-test harness.	Implementation gate: design outputs trace to requirements and risk controls.
Months 30-33	Verify, validate, stress-test, and feed learning back into GTF/CFM/AFE.	Verification report; validation report; stress-test report; learning-return record; revised RSM; public methods note.	Epistemic gate: evidence corroborates, weakens, falsifies, or revises the relevant axiom / requirement set.

7.3 Candidate Prototype Options

The final prototype should be selected only after the CFM requirements subset has matured. However, the following options are currently suitable because they test formation-accountability design without requiring a full AFE implementation.

Prototype Option	What It Tests	Why It Is Suitable
Formation-and-release interface	Whether outputs can arise, disclose conditions, remain provisional, transform, and dissolve.	Best all-around demonstration of formation-and-release lifecycle as a bounded engineering lifecycle.
Context-coherence monitor	Whether a system preserves relational context rather than drifting into isolated linear memory.	Directly addresses context drift and supports measurable coherence metrics.
Reification-detection module	Whether the system can detect when process/relational information is presented as fixed object reality.	Sharp prototype for non-reification and practical non-reification.
Formation-dynamics visualizer	Whether field, boundary, resonance, formation, memory, recognition, and release can be shown as a visual model.	Useful for collaborators, education, and stakeholder communication.
Dialectical stress-test harness	Whether candidate axioms and requirements survive rival explanations and failure modes.	Strongest prototype for avoiding tautological validation.

7.4 Post 33-Month Horizon: From Vertical Slice to Reference Architecture

After the bounded proof-of-method, IFII can decide whether the evidence justifies a broader AFE reference architecture. The next phase should generalize only from validated vertical slices. It should not assume that one successful prototype proves the full GTF. Instead, each slice should identify which primitives, axioms, metrics, requirements, and design patterns are robust enough to be reused.

Horizon	R&D Aim	Main Deliverable	What Is Being Tested
Reference architecture v1	Generalize from multiple validated vertical slices.	AFE reference core model, APIs, governance model, design pattern library.	Whether a reusable field-informed architecture is possible.
Ecosystem expansion	Train responsible builders and reviewers.	Architecture of Reality curriculum, safe-use guidance, community review forum.	Whether citizen engineers can extend the method without category errors or misuse.

C. IFII's Engineering Program and Preliminary Development Plan

Horizon	R&D Aim	Main Deliverable	What Is Being Tested
Standards and public-good infrastructure	Make the work reusable and auditable.	Open schemas, traceability templates, validation metrics, documentation.	Whether formation accountability can become a shared engineering practice.
Long horizon	Move beyond software-only prototypes where justified.	Analog formation spaces, field technologies, detector intelligence, visual intelligence research.	Whether formation-aware engineering has relevance beyond digital interfaces.

7.5 Success Criteria and Non-Success Criteria

The preliminary roadmap should be evaluated by proof-of-method criteria, not by premature product criteria.

Success Would Mean	Non-Success Or Warning Sign
One source subset, primitive constellation, axiom set, requirements subset, prototype, and evidence package form an auditable chain.	The project produces impressive outputs but cannot show source-to-requirement-to-evidence traceability.
The prototype visibly reduces opacity, reification, context drift, and capture tendencies in a bounded scenario.	The prototype behaves like a generic AI interface with spiritualized language added on top.
Human review gates are enforced for source fidelity, axiom status, requirements baseline, validation conclusion, and public release.	AI assistance begins to self-authorize source meaning, axiom acceptance, or validation conclusions.
Release, revision, rollback, forgetting, and archival work as technical functions.	Release remains a value statement rather than an implemented architecture feature.
Learning return changes the primitive registry, axiom version, requirements baseline, or prototype design when evidence requires it.	Validation confirms what IFII already believed without exposing the theory to correction.