Knowledge-Native AI: A Systems Vision for Computable Knowledge

Naveen Ashish
KnaiTai Open Initiative
nashish@knaitai.org

Abstract

Recent advances in generative AI have dramatically expanded the range of tasks machines can perform through language. Yet many consequential applications - including healthcare, engineering, scientific discovery, emergency response, and autonomous systems - depend on far more than language generation alone. They require continual interaction with structured knowledge, predictive models, domain-specific data, human expertise, and mechanisms for verification and governance. Modern AI systems increasingly need these capabilities to operate as a coherent whole rather than as loosely coupled external components.

This paper argues that AI is entering a new systems phase in which **computable knowledge becomes a first-class computational medium**. In Knowledge-Native AI, knowledge is not merely represented or retrieved; it is assimilated, scoped, composed, operationalized, verified, and evolved as the system operates. Rather than proposing another knowledge representation, ontology, or retrieval architecture, we present a systems perspective that coordinates heterogeneous knowledge sources, generative models, structured prediction models, human interaction, and symbolic reasoning.

We introduce four foundational abstractions - **Knowledge Spaces, Knowledge Capsules, Knowledge Functions, and Knowledge Conversations** - and examine their implications for AI systems engineering. We then outline an open research agenda and describe an open-source initiative for exploring these ideas in practice.

1. Introduction

The remarkable success of foundation models has transformed artificial intelligence. Systems capable of generating coherent language, software, images, and increasingly sophisticated explanations have expanded both the scope and accessibility of AI. Generative models are rapidly becoming the primary interface through which people interact with intelligent systems. This success has also influenced how AI applications are engineered. Language models are commonly augmented with retrieval, external tools, APIs, and agent frameworks, allowing them to access information and capabilities beyond their learned parameters. For many applications, this combination is remarkably effective.

As AI moves beyond general-purpose assistants toward medicine, engineering, scientific discovery, emergency response, autonomous systems, and other high-consequence domains, a more complex picture emerges. Language generation is only one component of a larger process. Effective operation depends on continual interaction with structured knowledge, predictive models, external data, human expertise, domain constraints, and explicit mechanisms for verification and governance.

Figure 1 illustrates the motivation using a simplified after-hours pediatric care scenario. Both workflows use a generative model to interact naturally with a caregiver. The difference lies not in the conversational interface, but in how knowledge participates in the interaction. In the upper workflow, the model must interpret the conversation, determine what information to collect, maintain context, and generate recommendations largely on its own. In the lower workflow, domain knowledge continually guides the exchange - shaping the questions asked, the interpretation of responses, the recognition of risk, and the validation of recommendations.

Figure 1. After-hours Clinical Care AI Agent

Several important observations emerge from this comparison.

- **Knowledge participates across the interaction**, not only during retrieval or final-answer generation.
- **The conversation itself becomes knowledge-guided.** Knowledge determines what information is still missing, which follow-up questions should be asked, and when sufficient evidence has been accumulated to proceed.
- **Patient context becomes computational rather than merely conversational.** Age, weight, medical history, medications, allergies, prior interactions, and caregiver responses become structured state that influences subsequent reasoning.
- **Knowledge provides operational guidance.** Clinical pathways, dosing rules, contraindications, red-flag symptoms, and escalation criteria shape system behavior rather than appearing only as reference material.

- **Safety and trustworthiness are supported across the workflow.** Knowledge enables verification, consistency checking, traceability, and appropriate escalation before recommendations reach the caregiver.
- **The model and the knowledge infrastructure complement one another.** The model enables natural interaction; computable knowledge provides structure, continuity, guidance, and operational constraints.
- **The underlying pattern is domain-independent.** Although illustrated through pediatric care, similar interactions arise in engineering, emergency response, logistics, behavioral analysis, robotics, scientific discovery, and other knowledge-intensive systems.

Figure 2 shows that the same underlying pattern appears in systems that interpret evolving behavior and support decisions under uncertainty. Such applications arise in transportation and logistics, supply-chain operations, emergency management, infrastructure monitoring, environmental systems, and many other settings driven by continuously changing observations.

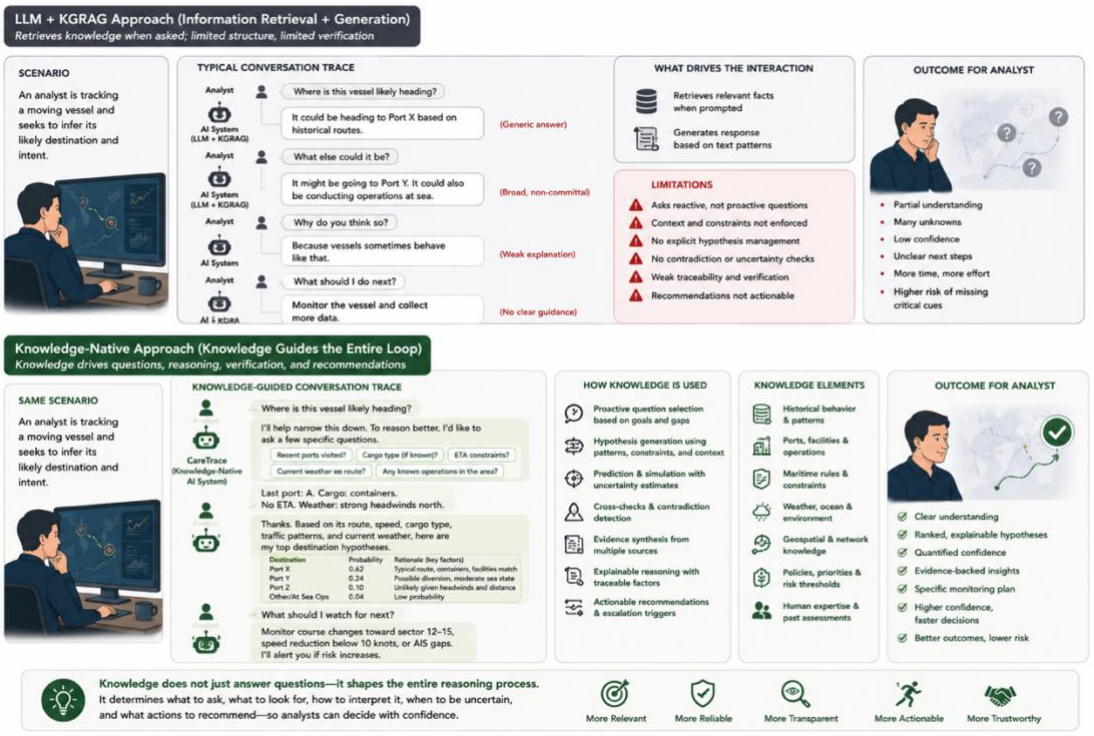

Figure 2. Dynamic Behavior Understanding

A conventional workflow can answer individual questions effectively. An analyst asks what an observed entity is likely to do, the model retrieves relevant information, and it generates a hypothesis or explanation. The interaction nevertheless remains largely reactive: knowledge is consulted in response to prompts rather than used to organize the reasoning process. As observations accumulate and competing explanations emerge, the system may offer limited support for identifying missing evidence, maintaining explicit alternatives, reconciling contradictions, or deciding when confidence is sufficient for action.

The lower workflow illustrates a different model. Computable knowledge participates throughout reasoning rather than merely supplying background information. Historical behavior, environmental conditions, infrastructure, operational constraints, policies, and human expertise jointly shape the interaction. They influence which questions should be asked, what observations would reduce uncertainty, which hypotheses remain plausible, how new evidence changes prior conclusions, and when a recommendation should be revised or verified.

Several observations emerge from this comparison.

- **Knowledge guides reasoning rather than merely answering questions.**
- **Reasoning becomes iterative and evidence-driven.** New observations refine competing hypotheses instead of producing isolated predictions.
- **Knowledge identifies what is still missing.** The system can expose evidence gaps and identify the next useful observation or question.
- **Verification becomes part of the reasoning loop.** Predictions, explanations, and recommendations are checked against domain knowledge, operational constraints, and new evidence.
- **Uncertainty is represented explicitly.** The system can preserve alternatives, supporting evidence, key unknowns, and confidence rather than prematurely selecting one explanation.
- **Human expertise remains integral.** People contribute observations, challenge assumptions, resolve ambiguity, and guide exploration where automated reasoning is insufficient.
- **The pattern generalizes across domains.** The sources of knowledge differ from those in Figure 1, but the interaction among models, computable knowledge, evidence, and human judgment remains structurally similar.

Figures 1 and 2 depict very different applications, yet expose the same systems pattern. In both cases, knowledge does more than supply facts. It guides information gathering, maintains relevant context, constrains interpretation, supports verification, manages uncertainty, and helps determine appropriate action. The particular knowledge differs by domain; its operational role remains strikingly similar.

These examples illustrate a broader transition occurring across modern AI. Knowledge has long been represented through expert systems, knowledge bases, ontologies, semantic web technologies, databases, and knowledge graphs. More recently, foundation models have demonstrated remarkable capabilities for acquiring implicit knowledge through large-scale learning. Collectively, these advances have transformed how AI systems acquire, represent, retrieve, and reason with knowledge.

Knowledge increasingly spans structured and unstructured sources, predictive models, human expertise, external services, policies, and evolving system state. It is no longer confined to a repository or retrieval layer. The emerging challenge is one of **knowledge operation**: how heterogeneous forms of knowledge are assimilated, scoped, composed, applied, verified, and evolved as part of a coherent AI system.

We call this systems perspective **Knowledge-Native AI**. It treats computable knowledge as a first-class computational medium through which models, data, services, policies, and people cooperate. The objective is not to introduce another ontology language, graph database, retrieval architecture, or reasoning formalism, but to identify the abstractions required to make knowledge operational across an entire AI system.

The defining characteristic of Knowledge-Native AI is not what knowledge a system possesses, but how that knowledge participates throughout computation.

The remainder of the paper traces the evolving role of knowledge in AI systems, develops the idea of knowledge as a computational medium, proposes foundational abstractions for Knowledge-Native AI, examines the implications for systems engineering, and outlines an open research and development agenda.

2. The Evolving Role of Knowledge in AI Systems

Knowledge has occupied a central role in artificial intelligence from its earliest systems. Successive research traditions have expanded both the forms knowledge can take and the operations that can be performed over it. Expert systems encoded domain expertise as executable rules. Knowledge bases organized structured facts for querying and inference. Ontologies and description logics introduced formal semantics and consistency. Semantic Web technologies emphasized interoperability across distributed sources, while knowledge graphs enabled flexible integration of heterogeneous entities and relationships. Foundation models subsequently demonstrated that broad knowledge could also be acquired implicitly through large-scale statistical learning.

These developments remain foundational. They made knowledge representable, executable, shareable, inferable, searchable, and increasingly scalable. More recent retrieval-augmented systems have further enabled generative models to access external information during execution. Knowledge-Native AI builds on this lineage rather than replacing it.

Today's AI systems increasingly operate under a different computational paradigm. Figures 1 and 2 illustrate two seemingly unrelated applications: an after-hours pediatric care assistant and a behavioral decision-support system. Although the domains differ dramatically, both systems exhibit a remarkably similar computational structure. Knowledge is no longer confined to a dedicated knowledge base or retrieval subsystem. Instead, it participates continuously throughout the system's operation. It guides what information should be collected, identifies missing evidence, constrains possible interpretations, supports verification, personalizes recommendations, explains decisions, and helps determine appropriate next actions. This represents more than an increase in the amount of knowledge available to an AI system. Rather, it reflects a fundamental change in **the role that knowledge plays during computation**. Knowledge no longer simply answers questions posed by the system; it increasingly helps determine **which questions should be asked, what computations should occur next, and how the overall reasoning process should proceed**.

Several observations characterize this transition.

- **Knowledge is increasingly active rather than passive.** It continuously influences system behavior instead of serving solely as information to be retrieved.
- **Knowledge becomes distributed across the entire system.** Structured knowledge, learned models, external data sources, operational constraints, policies, predictive models, and human expertise all contribute to computation.
- **Knowledge evolves continuously during execution.** New observations, user interactions, predictions, and feedback modify the knowledge available to subsequent reasoning.
- **Knowledge participates in decision making rather than simply supporting it.** It guides hypothesis generation, information gathering, verification, explanation, and recommendation throughout the computational workflow.
- **Knowledge operates at multiple levels of abstraction, simultaneously.** Domain knowledge, contextual knowledge, operational policies, historical observations, learned representations, and human expertise collectively shape system behavior.

Taken together, these observations suggest that modern AI systems are confronting a new systems challenge. The problem is no longer solely how to represent knowledge, nor even how to retrieve or reason over it efficiently. Instead, the challenge is how to design AI systems in which heterogeneous forms of computable knowledge operate coherently throughout system execution.

This shift bears an interesting resemblance to previous transitions in computer systems. Databases did not merely provide persistent storage; they transformed data into a first-class computational abstraction with associated runtimes, query languages, optimization techniques, and programming models. Distributed systems similarly elevated communication, coordination, and consistency into explicit systems concerns rather than implementation details hidden within applications. We argue that AI systems are undergoing a comparable transition. As language models become increasingly capable conversational interfaces, knowledge itself is emerging as a first-class systems concern. Future AI systems will require architectures that support not only the representation and retrieval of knowledge, but also its assimilation, composition, evolution, verification, and operational use within computation. The emerging challenge is therefore not only how knowledge should be represented or retrieved, but how heterogeneous forms of knowledge can participate coherently throughout the operation of an AI system. Table 1 summarizes this evolution in the dominant role assigned to knowledge.

Table 1. The evolving role of knowledge in AI systems.

Era	**Primary Contribution**	**Dominant View of Knowledge**
Expert Systems	Explicit rules and reasoning	Knowledge as encoded expertise
Knowledge Bases	Structured storage and querying	Knowledge as organized facts
Ontologies & Semantic Web	Shared semantics and inference	Knowledge as formal representation
Knowledge Graphs	Connected, heterogeneous information	Knowledge as a graph of entities and relationships
Foundation Models	Implicit statistical knowledge	Knowledge embedded within model parameters

Retrieval-Augmented AI	Dynamic access to external information	Knowledge as an external resource
Knowledge-Native AI	Knowledge participates throughout system computation	Knowledge as a *first-class computational medium*

3. Knowledge as a Computational Medium

Knowledge has traditionally been treated as something to represent, organize, retrieve, or reason over. The emerging systems challenge requires a broader view: computable knowledge must participate throughout the acquisition of information, the formulation of hypotheses, the interpretation of evidence, and the selection and verification of actions. We propose viewing **computable knowledge as a computational medium** rather than simply as a computational resource. The distinction is subtle but profound.

Modern AI systems increasingly compute through knowledge, rather than merely with access to knowledge.

Relational databases provide an instructive analogy. A database is far more than persistent storage. Data participates continuously in application execution through indexing, query optimization, transaction management, integrity constraints, concurrency control, and views. Likewise, modern operating systems transformed files from passive storage objects into computational abstractions supporting sharing, synchronization, persistence, and composition. In both cases, the abstraction extended well beyond representation. We argue that computable knowledge is beginning to undergo a similar transition.

This distinction is illustrated by the examples presented in Figures 1 and 2. In both applications, knowledge continuously influences system behavior. It determines what additional information should be collected, constrains the space of plausible interpretations, identifies missing evidence, verifies intermediate conclusions, explains recommendations, personalizes reasoning, and governs appropriate next actions. These operations across the workflow, rather than appearing as isolated retrieval steps. This perspective differs fundamentally from treating knowledge as an external repository consulted only when additional information is required. Instead, knowledge continuously interacts with language models, structured prediction models, external data sources, operational policies, human expertise, and evolving system state. The resulting computation is collaborative rather than sequential. Knowledge no longer occupies a single architectural component; instead, it permeates the system as a whole (Figure 3).

Several characteristics distinguish knowledge operating as a computational medium.

- **Knowledge is assimilated rather than simply accessed.** AI systems continuously accumulate information from user interactions, observations, predictive models, external services, and human expertise.
- **Knowledge is scoped rather than merely retrieved.** The system dynamically determines which subset of available knowledge is relevant to the current computational objective while maintaining awareness of broader context.
- **Knowledge guides computation rather than simply supporting it.** It determines what information remains necessary, which questions should be asked next, which hypotheses deserve further investigation, and which computational pathways remain feasible.

- **Knowledge verifies computation.** Intermediate reasoning, predictions, recommendations, and generated responses are continually evaluated against domain knowledge, operational constraints, and accumulated evidence.
- **Knowledge evolves during execution.** New observations, user feedback, external events, and validated conclusions continually modify the knowledge participating in subsequent computation.
- **Knowledge governs system behavior.** Policies, regulations, organizational objectives, safety constraints, and operational procedures become executable computational participants rather than passive documentation.

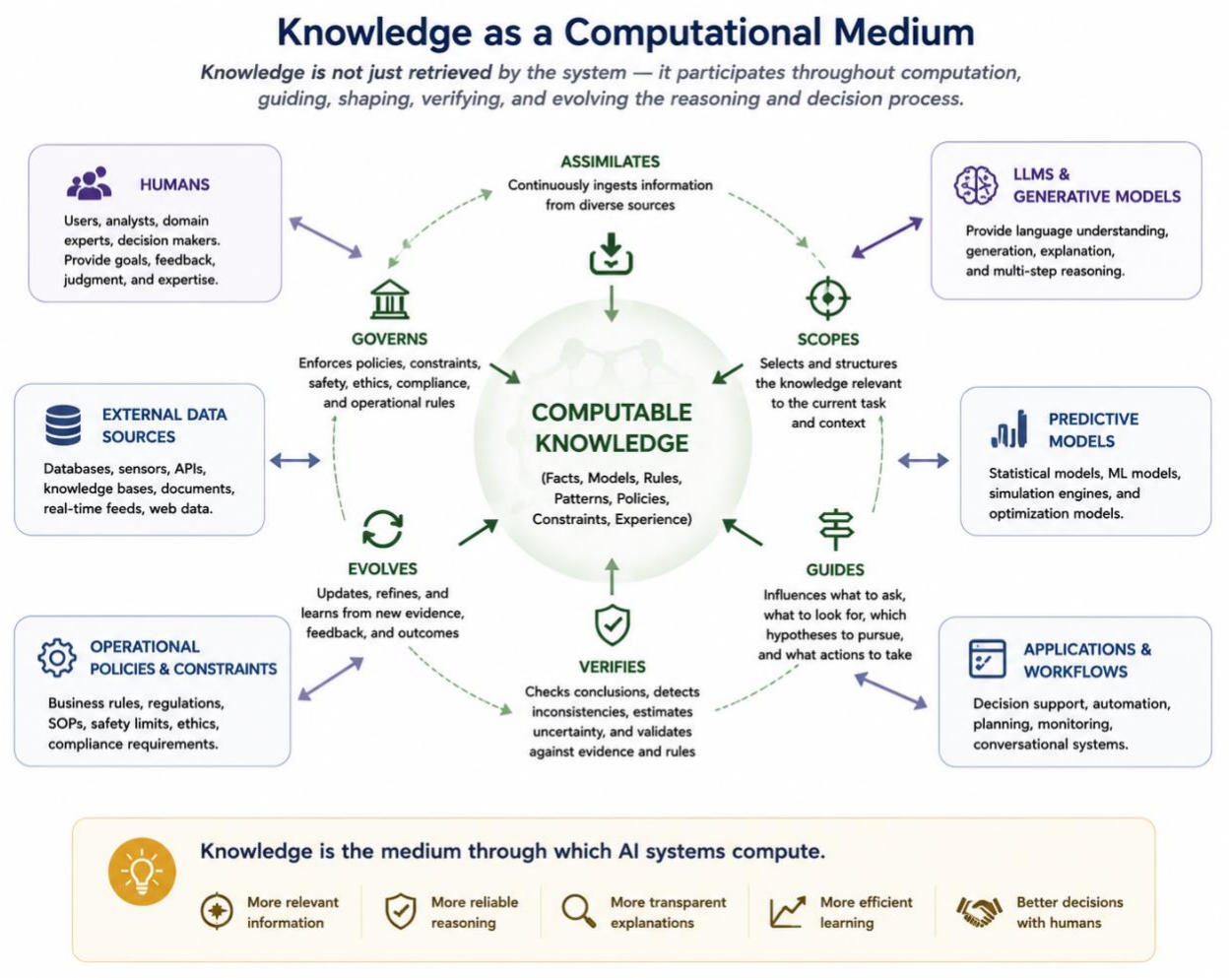

Figure 3. Knowledge as a Computational Medium

These capabilities make knowledge an organizing substrate for interaction among generative models, predictive models, external information sources, operational policies, human participants, and applications. This observation has important implications for AI systems engineering. If knowledge functions as a computational medium, then future AI systems require programming abstractions analogous to those introduced by operating systems, databases, and distributed computing. Developers must reason explicitly

about how knowledge is assimilated, scoped, composed, verified, evolved, and governed throughout computation, rather than treating these concerns as application-specific implementation details.

The next section develops a small set of programming abstractions for organizing systems around this principle.

4. Knowledge-Native AI

If computable knowledge is treated as a first-class medium, the resulting system must be organized differently from one in which knowledge appears only through retrieval, prompting, or post-processing. Knowledge-Native AI is **not** a particular model architecture, knowledge representation, graph database, ontology language, or retrieval strategy. Its defining feature is the organization of computation around the continuous interaction of models, data, knowledge resources, policies, services, and human participants. Individual components may vary, but their interaction is mediated through explicit, operational knowledge rather than application-specific glue alone.

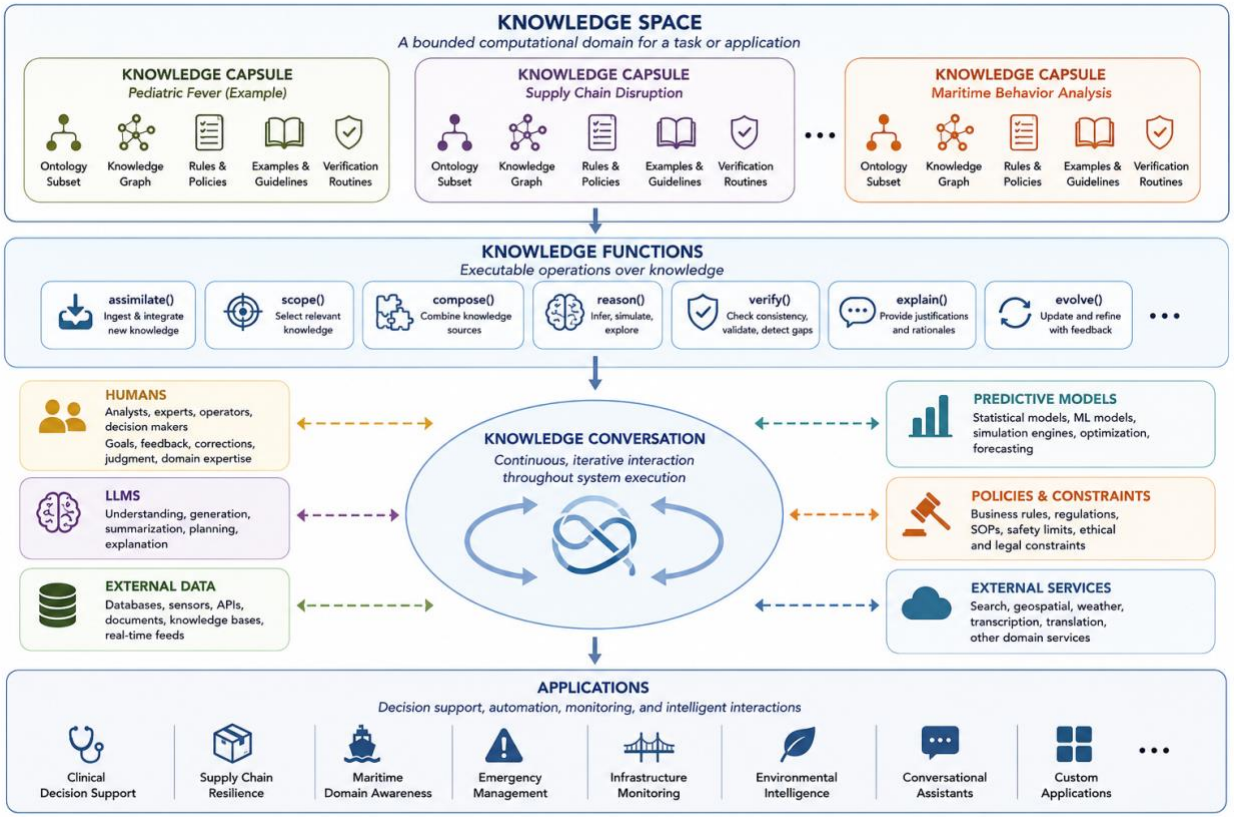

Figure 4. Programming Knowledge-Native AI Systems

This perspective requires abstractions that remain stable even as particular language models, databases, reasoning engines, or predictive models evolve. We propose four foundational abstractions.

- **Knowledge Spaces** define bounded domains of computable knowledge associated with a task, application, or evolving problem context.
- **Knowledge Capsules** package coherent and reusable subsets of knowledge together with the rules, policies, examples, models, and verification procedures needed to operate over them.
- **Knowledge Functions** expose high-level operations for assimilating, scoping, composing, reasoning with, verifying, explaining, and evolving knowledge.
- **Knowledge Conversations** coordinate the iterative exchange among generative models, predictive models, data sources, services, policies, human participants, and knowledge throughout system execution.

These abstractions are intentionally independent of any specific implementation technology. They describe **how developers reason about knowledge within AI systems**, rather than prescribing how that knowledge must be represented internally. Figure 4 summarizes this emerging programming model. Computation no longer proceeds through isolated interactions between an application and a knowledge repository. Instead, knowledge becomes the organizing medium through which language models, structured prediction models, external data sources, human expertise, and operational workflows continuously collaborate.

5. Implications for AI Systems Engineering

Viewing knowledge as a computational medium changes how knowledge-intensive AI software is engineered. Knowledge engineering can no longer be treated as an isolated activity concerned only with ontology construction, information retrieval, or graph management. It becomes a systems concern affecting modularity, composition, verification, scalability, deployment, and long-term maintenance. This distinction becomes increasingly important as AI systems evolve from isolated demonstrations into complex, continuously operating software systems. While today's language models provide extraordinary capabilities for language understanding and generation, developers remain responsible for integrating domain knowledge, external data sources, predictive models, operational constraints, organizational policies, human expertise, and verification mechanisms into coherent applications. As these systems grow in complexity, the principal engineering challenge shifts from simply connecting components toward managing how knowledge flows across the system.

Several engineering implications naturally emerge from this perspective.

5.1 Progressive Knowledge Engineering

Knowledge-intensive AI systems rarely begin fully formed. Developers typically start with narrowly scoped capabilities, validate system behavior, and progressively expand both functional scope and knowledge coverage. A clinical assistant may initially support a single disease before expanding to an entire specialty and eventually to primary care. Similarly, a transportation or logistics application may begin with a single operational scenario before incorporating additional infrastructure, regulations, environmental conditions, and behavioral models. Knowledge-Native AI encourages this style of incremental development by treating knowledge as modular computational assets that can evolve independently of the surrounding application.

5.2 Knowledge Becomes Modular Modern software engineering emphasizes modularity through reusable software components. A similar principle applies to computable knowledge. Rather than constructing a single monolithic knowledge repository, developers should be able to organize knowledge into coherent computational units that can be independently developed, validated, versioned, reused, and composed. Different applications may share common knowledge while introducing domain-specific extensions without modifying the entire system. This modularity becomes increasingly important as AI systems integrate multiple knowledge representations, predictive models, and external services.

5.3 Knowledge Engineering Becomes Systems Engineering Historically, knowledge engineering focused primarily on acquiring and representing domain expertise. Modern AI systems introduce a broader set of engineering concerns. Developers must determine:

- how knowledge is assimilated from multiple heterogeneous sources,
- how relevant knowledge is identified for a particular computational objective,
- how knowledge interacts with language models and predictive models,
- how intermediate reasoning is verified,
- how knowledge evolves through human feedback and operational experience,
- and how these processes remain efficient as systems continue to grow.

These questions extend beyond representation. They become systems engineering problems.

5.4 System Operation Becomes Knowledge-Centric Traditional AI applications frequently organize computation around model inference or information retrieval. Knowledge-Native AI instead organizes computation around the continual interaction between knowledge, models, data, and human expertise. Language models remain essential conversational interfaces, but they become participants within a broader computational workflow rather than the sole driver of system behavior. Knowledge determines not only what information is available, but also how computation proceeds. This perspective naturally supports richer interactions involving explanation, verification, uncertainty management, policy enforcement, and human oversight.

5.5 Trustworthiness is Engineered into the System Many trustworthy AI techniques operate after model inference through explanation, rule checking, or post-hoc validation. Knowledge-Native AI instead enables trustworthiness to emerge throughout system execution. Verification, constraint checking, evidence accumulation, policy compliance, contradiction detection, and human review become continuous computational processes rather than isolated downstream activities. Trustworthiness therefore becomes a system property that must be designed into the workflow, rather than an auxiliary capability applied after model inference.

6. The Road Ahead

The abstractions introduced above define a broad research agenda. Knowledge Spaces, Capsules, Functions, and Conversations raise foundational questions spanning artificial intelligence, databases, programming languages, software engineering, human-computer interaction, and knowledge representation. Developing these ideas into a mature systems discipline will require advances in how

knowledge is acquired, transformed, optimized, verified, evolved, and collaboratively maintained. Several directions appear especially important.

Knowledge Assimilation How should AI systems continuously acquire, organize, validate, and integrate knowledge originating from documents, structured databases, knowledge graphs, predictive models, simulations, external services, and human experts? Unlike traditional data ingestion pipelines, knowledge assimilation must preserve semantics, provenance, uncertainty, and operational context while remaining computationally efficient.

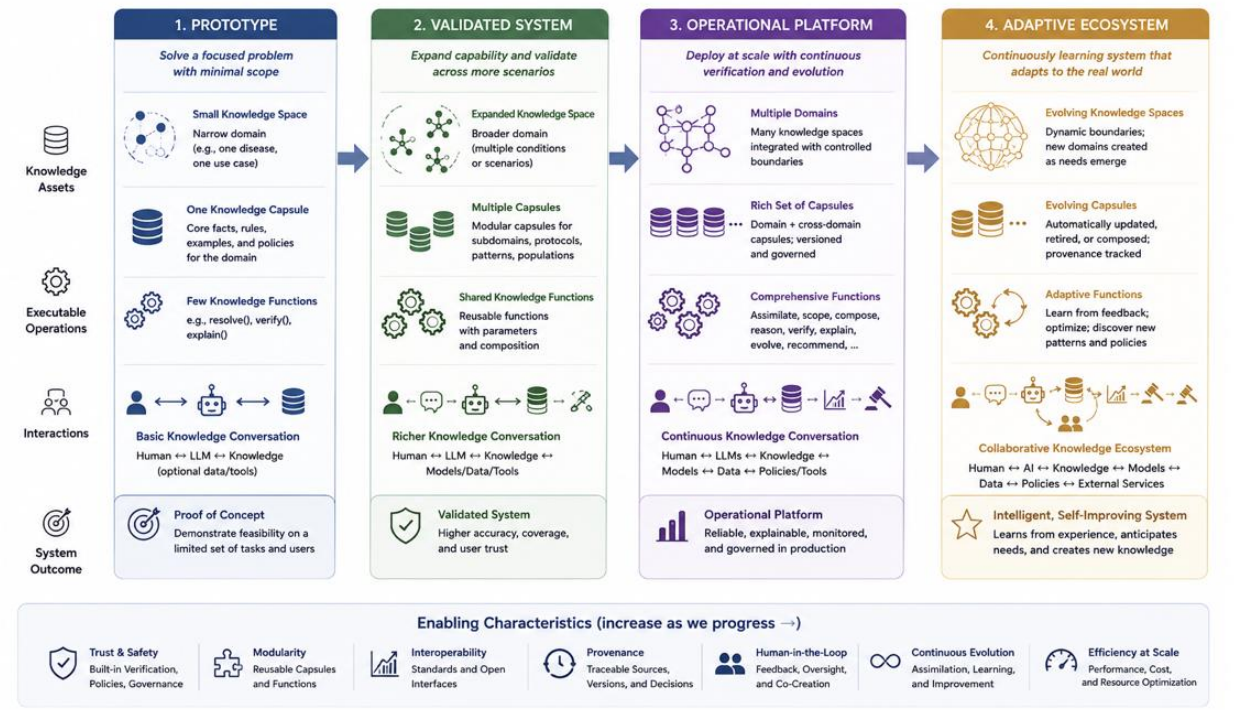

Figure 5. Progressive Engineering of Knowledge-Native AI Systems

Knowledge Compilation Modern software relies on compilers to transform high-level programs into efficient executable systems. An analogous capability may be required for knowledge. Large heterogeneous knowledge sources must often be transformed into compact, application-specific knowledge capsules that support efficient reasoning, verification, and execution while preserving correctness and traceability.

Knowledge Algebra As knowledge becomes a computational abstraction, developers require principled operations for manipulating it. Questions naturally arise regarding the composition, decomposition, specialization, extension, intersection, versioning, and evolution of knowledge spaces and knowledge capsules. We believe that a formal knowledge algebra represents an important long-term research direction for Knowledge-Native AI.

Knowledge Optimization Not all available knowledge should participate equally in every computation. AI systems must determine which knowledge is relevant, when it should become active, how much knowledge should participate, and how competing sources should be reconciled. These problems resemble

query optimization in database systems, yet operate over heterogeneous knowledge rather than structured data alone.

Knowledge Verification As knowledge increasingly guides system behavior, verification becomes a continuous computational activity rather than a final validation step. Future systems must verify not only generated responses, but also intermediate reasoning, knowledge consistency, policy compliance, supporting evidence, uncertainty estimates, and operational constraints throughout execution.

Knowledge Evolution Knowledge-intensive AI systems will continuously learn from new observations, user interactions, corrections, operational experience, and scientific discovery. Supporting safe, explainable, and traceable knowledge evolution represents a significant systems challenge extending beyond traditional model retraining or database updates.

Human–Knowledge Collaboration Human expertise remains essential in many knowledge-intensive domains. Rather than replacing domain experts, Knowledge-Native AI seeks to integrate human judgment directly into the computational process. Future systems should support collaborative knowledge construction, interactive verification, incremental refinement, and transparent decision making in which humans and AI systems continuously contribute to shared computational knowledge.

Taken together, these directions suggest that Knowledge-Native AI extends beyond a single software framework or implementation. They point toward a broader ecosystem comprising programming models, runtime systems, development environments, optimization techniques, verification methodologies, educational resources, benchmark problems, and open-source software. Progress will require contributions from artificial intelligence, databases, software engineering, programming languages, human-computer interaction, distributed systems, knowledge representation, and numerous application domains. For this reason, we view Knowledge-Native AI not as a finished technology, but as an emerging systems discipline whose full implications remain to be explored.

Table 2. Illustrative analogies between classical systems abstractions and Knowledge-Native AI.

Classical Systems	Knowledge-Native AI
Compiler	Knowledge Compilation
Query Optimizer	Knowledge Optimization
Type System	Knowledge Verification
Runtime	Knowledge Conversation
Database	Knowledge Space
Library	Knowledge Capsule

7. The KnaiTai Open Initiative

Realizing Knowledge-Native AI will require more than a single algorithm or software package. It will require programming abstractions, reusable infrastructure, reference implementations, educational

resources, benchmarks, and sustained experimentation across real applications. **KnaiTai**[1] is an open-source initiative created to support that effort. It provides a shared environment for exploring how computable knowledge can become a native participant in AI systems. KnaiTai is intended to complement - not replace - language models, knowledge graphs, databases, reasoning engines, agent frameworks, and existing software ecosystems. Several complementary activities will evolve together within the initiative.

- **Open Infrastructure.** Reusable components for creating, composing, validating, and executing knowledge-native systems.
- **Programming Model.** A coherent vocabulary and set of interfaces centered on Knowledge Spaces, Capsules, Functions, and Conversations.
- **Research Platform.** A common environment for experimentation in assimilation, compilation, algebra, optimization, verification, and evolution.
- **Reference Applications.** Progressively richer implementations that test the abstractions across distinct knowledge-intensive domains.
- **Education and Community.** Open documentation, tutorials, programming examples, reusable knowledge assets, collaborative projects, and the planned developer guide, *Programming Knowledge-Native AI*.

KnaiTai will evolve incrementally. Initial implementations will begin with narrowly scoped Knowledge Spaces and a small number of Capsules and Functions, then expand through validation in progressively richer applications. The aim is not to specify an entire ecosystem in advance, but to allow the abstractions, software, and development practices to mature together through open experimentation. KnaiTai is therefore not presented as the completed realization of Knowledge-Native AI. It is an open foundation for discovering what such systems should become.

8. Conclusion

Language models have dramatically expanded the range of problems machines can address. Yet many consequential AI applications require knowledge that guides information gathering, reasoning, prediction, verification, decision making, and human collaboration, as the system operates. This paper has argued that this shift calls for a new systems abstraction: computable knowledge as a first-class computational medium. Knowledge-Native AI organizes systems around this principle through Knowledge Spaces, Knowledge Capsules, Knowledge Functions, and Knowledge Conversations. Much remains to be developed and tested. Enduring advances in computing, however, often begin with abstractions that allow complex systems to be understood and built differently. We hope the KnaiTai open initiative contributes to the research, engineering, and shared experimentation needed to make knowledge native to the next generation of AI systems.

[1] https://knaitai.org