# ATLAS

## A Question Oriented Approach to the Use of Pattern Languages in Knowledge Management

**Contributors:**

R.J. Cordes

Scott David

Daniel Friedman

Alexandra Mikhailova

Andrew Penland

Sam Young

Colten Zacharias

# Abstract

---

The ATLAS system, evolving since the late 1990s, stands as a dynamic and comprehensive knowledge management tool that intends to address the complexities of modern information supply chains. The antecedent to ATLAS was the Atlas of Risk, an informal assemblage of various risks associated with digital interactions. Here we provide an initial specification for digital prototypes and paper-and-pencil implementations of a matured ATLAS architecture which integrates pattern language approaches with question-oriented procedures to manage and interpret meaning and context. The ATLAS system facilitates the management and communication of nuanced data sets and knowledge bases with an eye towards interoperability without the need for fully shared standards. The development of ATLAS, driven by the need for enhanced data interoperability and shared understanding in an increasingly complex and volatile digital landscape, reflects a profound, community response to the challenges of information environments and the fragility of extreme specialization. ATLAS's ongoing evolution showcases its adaptability and significance in the realms of data analysis, knowledge management, and cognitive security, and this first release of a technical specification establishes a foundation for a transition from prototype to scale-appropriate implementation.

# Table of Contents

# Introduction

The modern information supply chain is struggling to keep pace with the explosions in volume and technical complexity of available information caused by its own innovations and advancement. From the time it was possible for a single human being to possess a large percentage of the knowledge in their immediate cultural context, to present day, our sociotechnical infrastructure for advancing and sharing knowledge has been in a positive feedback loop, compensating for the increasing rate of information production and the ever-widening gap between available information and the limits of individual situational awareness, capability, and memory. This general sociotechnical infrastructure, or more generally, synthetic intelligence, is primarily composed of individual and organizational specialization and expertise augmented by (i) knowledge, data, and narrative information management technologies (e.g., libraries, notetaking and archiving methodologies, standards, and databases) and (ii) generative and evaluative information processes and tools (e.g., the scientific method, mathematics). However, in any complex adaptive system, specialization comes at the cost of plasticity, or the ability to adapt to new circumstances.

In biology, this concept of trade-offs related to specialization is sometimes referred to as the "fragility of extreme specialization", and in engineering as "efficiency *is* fragility". In terms of the information supply chain, the diminishing returns of specialization are found in the proverbial mountains of redundant and obsolete work caused, in part, by the innumerable information silos generated by both knowledge specialization and the limits of human situational awareness [1]. Each year, millions of peer-reviewed papers, preprints, reports, and data sets are published, with valuable insights left unread and siloed across myriad domains. Even organizations composed of individuals at the highest levels of competence are inevitably left to manage the trade-offs among (i) filtering out, purposefully or otherwise, large swathes of potentially valuable information, (ii) adapting to a constant state of information overload and including large swathes of low-quality information, and (iii) externalizing analysis to generative and other statistical models, which carries a set of risks so diverse that it would exhaust the scope of any summary.

Information management systems, already struggling with factors such as provenance, addressability, privacy, standardization, quality, and explainability are now threatened by the emergence of highly accessible generative tools, such as Artificial Intelligence (AI) and Large Language Models (LLMs), which will further flood this already overly rich information space with the potential to bury more insight than they reveal. While modern information supply chains manage the logistics of *data* reliably and at scale*,* similar attention now needs to be applied to the logistics of *meaning*. Mechanisms for generating and sharing coherent maps of specialized information silos and their intersections are necessary to maintain navigability of the information environment.

## Atlas of Risk

While the laws of physics and computing that support purely technical interoperability and communications are global (and indeed universal), the laws of people and social and cultural norms are not. As the standardized Internet technologies interconnected across the globe, the flows of data made possible by the technologies encountered barriers from non-technical network variables. For example, the laws of certain jurisdictions prohibited access to certain types of content, and intellectual property rights (such as under copyright, patent, trademark and trade secret laws) also constrained the free flow of data and interactions. Further, notions of intrusions on personal privacy of individuals invited specific forms of protection and limitations on free flows of data, as did proprietary interests of businesses and secrecy and security needs of governments. Emerging organically in response to the need to render new and emerging risks in information networks visible and measurable, was The Atlas of Risk.

The Atlas of Risk was originally compiled (in the late 1990s, by Scott David, a contributor to this document) as an informal checklist of non-technical definitions of common business and legal risks associated with online and digital interactions and commercial transactions that proliferated along with the highly distributed information infrastructure of the early Internet, helping communicate siloed knowledge [2]. Following growth of and feedback on this initial checklist, it began to mirror prior work on “Pattern Languages”, helping to make “patterns” of practices visible so that they could be considered as candidates for formalization and measurement. Pattern Language approaches, originally popularized by Christopher Alexander, are intended to enable deep compressions of subjective commonalities in terms of objects and phenomena in a given environment. Where Alexander’s work on the topic helped identify and communicate ontology, intentions, conditions, and approaches in the domain of architecture [3], the Atlas of Risk focused on information architecture and engagement.

### ATLAS Exploration in 2023

Growing from the original informal checklist, the core listing of risks for the Atlas was compiled in 2014 into a simple digital document and later, in 2016, into a slide deck format [2]. When the slide deck exceeded 1,700 slides, it was clear that a more scale-appropriate and accessible interface was needed. Building upon the foundational Atlas of Risk, contributors to this document recognized that the Atlas of Risk represented an emergent, next iteration of prior written pattern languages. Instead of snapshot publications of patterns from specific domains, representing patterns of interest as data structures within a dynamic living knowledge resource would allow for continuous development by relevant communities and exchange among them.

Work on a first digital prototype of an ATLAS (stylized in all upper-case) applied to common vulnerabilities and exploits (CVE) databasing related to Cognitive Security began in March 2023, resulting in an “alpha” version of the “COGSEC ATLAS” [4] built using the low-code data and knowledge management platform Coda.io. This alpha version did not offer full community interactive capacity, but it did inspire local contribution and extension of the resource, and the

research and development led to valuable insights related to Cognitive Security itself - mainly that numerous communities maintain pattern languages with limited interoperability, underscoring the necessity for enhanced integration approaches. More importantly, it revealed valuable insights about how to render pattern language components interoperable across domains, and what is possible given an appropriate data structure:

1. Using modular requests for information about a given pattern, which would resolve to some other object (e.g., what is the remedy to this pattern of risk?), (i) allows new patterns and objects to be generated and networked as a function of resolving requests, and (ii) makes community contribution simpler and more measurable, productive, and efficient (see Figure 1).
2. Giving patterns a flexible data structure (i.e., no set schema of potential attributes), allowed for a greater variety of patterns from a variety of domains to become interoperable in relation to the attributes they do share - and where attribute values and attributes themselves are shared, new patterns can be revealed. Further, whole new derivative ATLAS structures become possible, for example, (i) patterns of influence which came from the Narrative Campaign Field Guide [5], a snap-shot of patterns related to narrative influence campaigns, were easily rendered into a separate derivative ATLAS (see Figure 2), and (ii) examples of emergent patterns become immediately and easily filtered into their own provisional collections (see Figure 3).
3. Using a Reference ID system, which can be shared to establish common reference to objects even where data about those objects might not be agreed upon, allows for simple merging and disambiguation in sharing information among communities with ontology overlaps and conflicts.

By onboarding contributors from an adjacent working group researching the structure of and information latent within questions and integrating past work on question oriented documentation [6] the following was revealed:

1. Using a combination of flexible parent-child relationships among patterns (e.g., a pattern may be an example or child of another pattern) and lists of questions we would expect to ask of items which fit some pattern (”QKits”), a “dynamic type system” emerges, wherein objects are assigned implied Patterns as a result of being involved in a particular request for information either as the subject or the response (e.g., the question: “what is the national flag of this country?” reveals that the subject is a country, and the response, a flag).
2. By carefully structuring the requests for information, we can not only route information to fill multiple attribute values, but enable a variety of complex operations, including new question generation and continuous updating of visualizations (see Figure 3).

As a result, the ATLAS effectively became a structured question and pattern generator, allowing several dozen collections of patterns related to Cognitive Security and from other fields to be integrated rapidly, resulting in a collection of over 1,000 individual, networked patterns (see Figures 5 and 6). Its initial release and controlled distribution revealed additional insights related to reliability of information provision and accessibility, leading to a recognized need for special designations for ATLAS instances with operators capable of being responsible parties in a larger information supply chain.

This phase marked a significant step forward in our research methodology by integrating both established and emerging patterns, and led the contributors to create a fresh set of data structures intended to leverage the insights and theoretical foundations discovered through the first phase of development. In this document, we present the resulting first version of a specification for digital and paper-and-pencil implementation of a networked ATLAS, with the intent to expand its contents based on feedback and continued work in 2024.

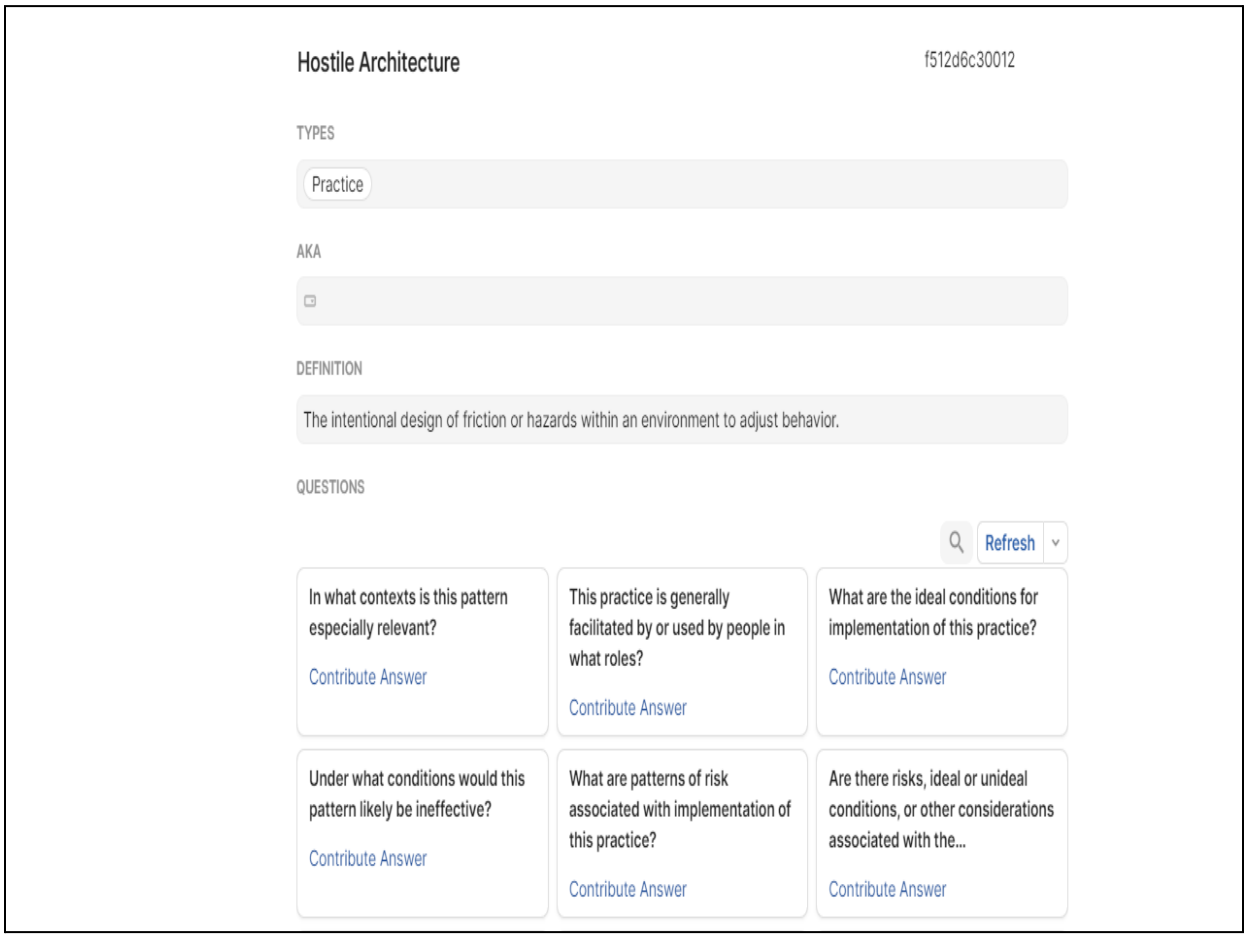


Figure 1. Example Pattern in the COGSEC Community ATLAS, with questions generated as a result of dynamic typing as a “Practice”.

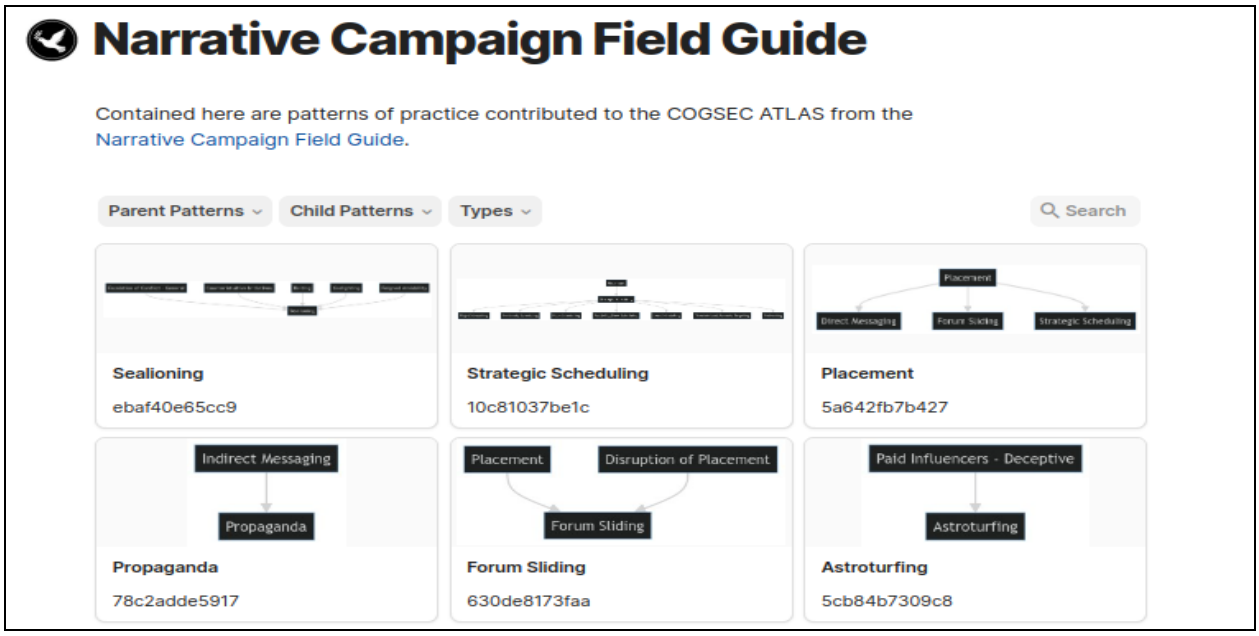


Figure 2. Snapshot of the Narrative Campaign Field Guide, rendered within ATLAS.

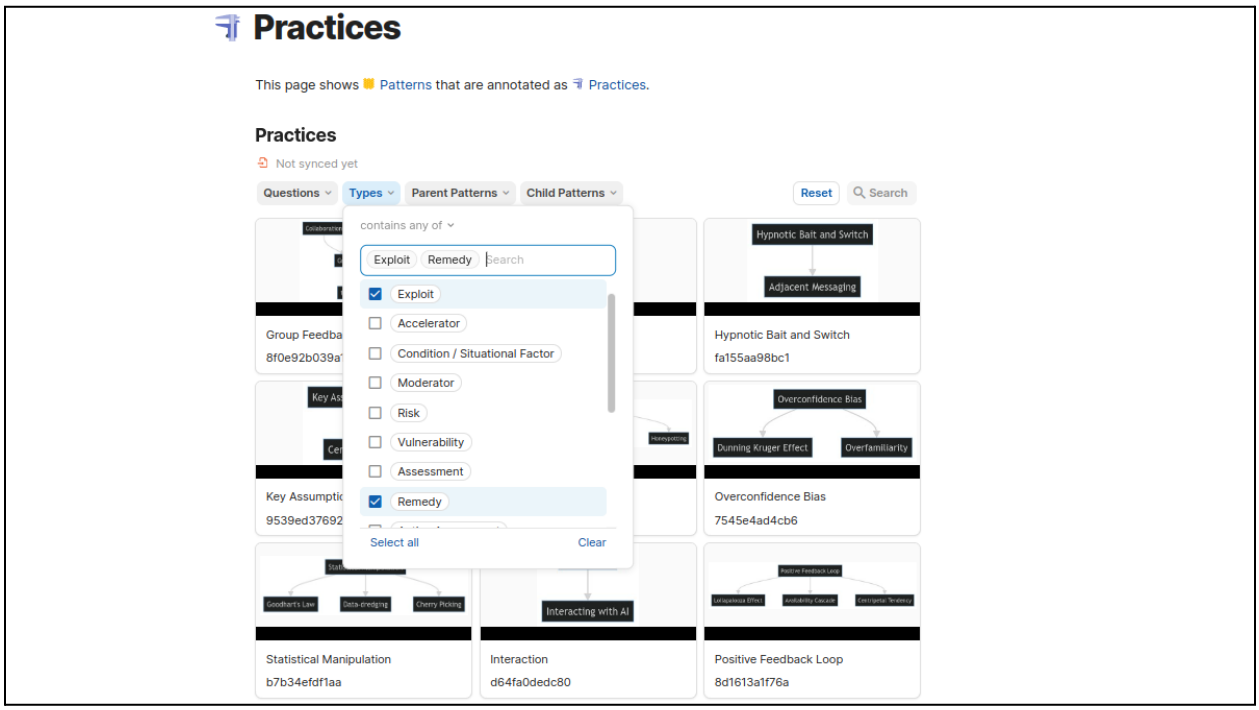


Figure 3. Snapshot of the ATLAS prototype page for patterns annotated as “Practices”. The patterns are further refined with the dropdown selection list, to patterns that have further been annotated as either Exploit or Remedy (these are practices that are either an exploit or remedy in a given setting).

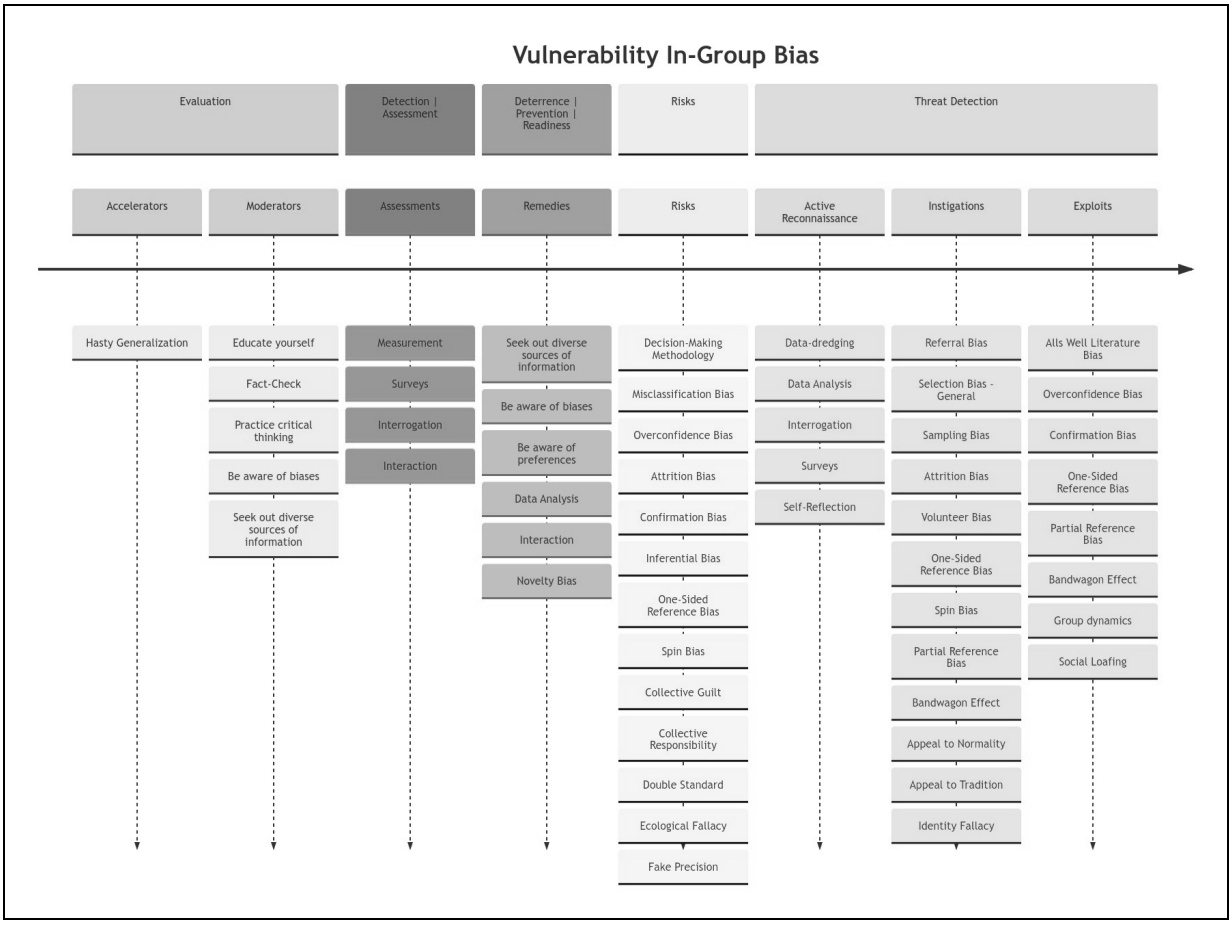


Figure 4. Snapshot of an automated re-rendering of attributes associated with a Pattern of Vulnerability in the COGSEC ATLAS. Each item was a response to a question which routed answers to an array, dynamically typing them and consequently generating new questions. Resulting visualization was referred to in the community as a “COGSEC ChillChain”.

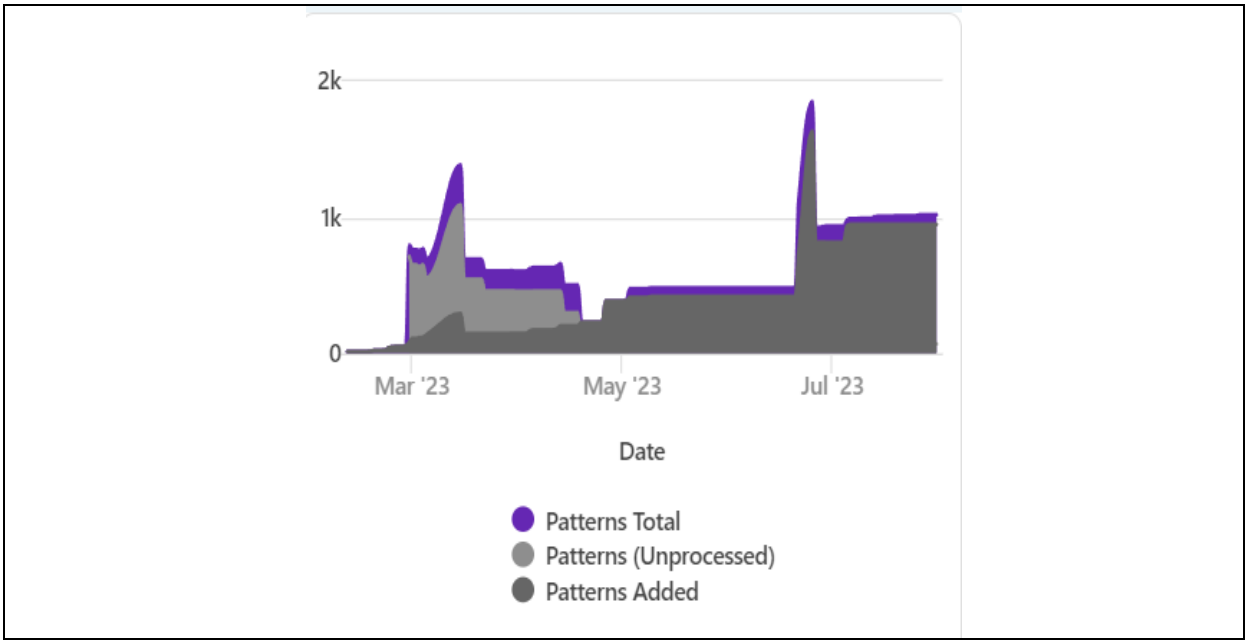


Figure 5. Time series of Pattern accumulation in the COGSEC Community ATLAS from time of initial alpha release in February to closing community access in August. Decreases caused by merging of duplicate entries.

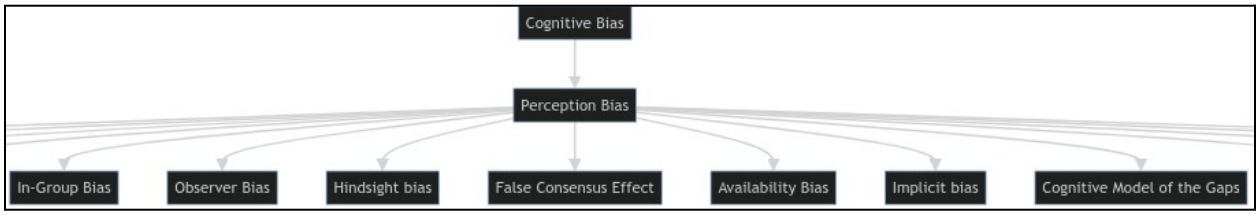


Figure 6. Snapshot of an automatic rendering of Parent-Child relationships of the focal pattern “Perception Bias”. Perception Bias is a child of the more general “Cognitive Bias”, and has the child patterns at bottom of the image.

# System Overview

## Purpose

ATLAS, as a Knowledge Management System, is intended to:

- enable the development and documentation of local and community information standards,
- facilitate rapid expansion and networking of knowledge,
- provide a basis for sharing common reference to objects (data, artifacts, abstractions) within, between, and among communities,
- reframe intercoder reliability issues as useful information, and
- restructure existing and developing knowledge bases to:
    - enable interoperability and exchange,
    - reveal knowledge gaps by default, and
    - support synthetic intelligence by providing a foundation for stable communication among human and automated agents, (e.g., by providing structured, streaming updates to artificial intelligence agents using retrieval augmented approaches).

## Scope

This document addresses the core components of the “ATLAS” system, a question-oriented approach for structuring, expanding, and establishing relationships within a knowledge base, and for dynamic typing of objects using pattern language approaches. The components, properties, and methods included would allow for functionality of the core mechanisms of the system both in digital prototype and paper-and-pencil implementations, similar to Zettelkasten, slip-box, and other similar knowledge management approaches [7, 8].

This version of the document only defines the necessary components, properties, and methods of the system which enable such paper-and-pencil implementation and pseudo-code, prototype implementations, it does not address:

- methods, schemas, or systems for digital Reference ID implementation,
- methods, properties, or implementation of interfaces and capabilities related to prompts (Prompt Interfaces),
- choice of programming languages or database software,
- technical implementation or supporting software and technologies, or
- data security and privacy.

Various features important for multi-user implementations of ATLAS system are not addressed here, such as:

- version control,
- governance, or
- explicit procedures for sharing data among independent ATLAS implementations.

These and similar aspects are left to designers, developers, and operators (DDOs) of ATLAS implementations.

## Intellectual Property & Licensing



## Additional Disclaimers

In the description of the ATLAS system in this document, the contributors assume no responsibility or liability for any errors or omissions. The information contained in this initial versioned release ATLAS 1.0 of the document is provided on an "as is" basis.

The authors and contributors makes no express or implied warranty as to the condition of any such information or materials, or as to the condition of any research or information generated under this version of the document, or as to any products made or developed under or as a result of this version of the ATLAS system including as a result of the use of information generated hereunder, or as to the merchantability or fitness for a particular purpose of such research, information, or resulting product, or that the information and materials provided will accomplish the intended results or are safe for any purpose including the intended purpose, or that any of the above will not interfere with privately- owned rights of others.

Neither the authors nor contributors shall be liable for special, consequential or incidental damages attributed to such information and materials provided under this version of the ATLAS SYSTEM or such research, information, or resulting products made or developed under or as a result of this version of the document.

## Structure of this Document

This remainder of this document consists of 4 sections:

**Theoretical Foundations.** A short summary of the theoretical foundations which were driving factors in the development of the definition of the ATLAS system.

**Systems Definition.** A system overview followed by a description of all core object classes and components, with details regarding their relationships, function, properties, and methods.

**Mechanisms.** A description of core mechanisms which are designed into or emerge from interactions among these components.

**Implications and Future work**. A description of implications of implementation of the system and recommendations for future work.

## Definitions, Acronyms, Abbreviations, and Word Usage

**API**: API is an acronym for Application Programming Interface. It is a set of protocols, routines, and tools that define how software components and systems should interact and communicate with each other, allowing for data exchange and integration.

**Array**: An array is a collection of objects.

**Boolean**: A Boolean refers to a data value which can either be True or False.

**Collision**: “Collision”, in this document, refers to conflicts in data assignment, for instance, where more than one attempt is made to set a data value and the assigned values are not equal.

**DDO**: An acronym for Designer, Developer, and/or Operator. Used in this document to refer to the organization, individual, or community which designs, develops, or operates an interface for interaction with an ATLAS instance.

**DMCA**: An acronym for The Digital Millennium Copyright Act, a United States copyright law addressing online service providers, copyright owners, and users of copyrighted materials.

**Dynamic Hypergraph**: A generalization of the mathematical object: graphs. Hypergraphs contain a set of vertices (representing abstract objects) and a set of hyperedges (connections between any number of vertices). Graphs are dynamic if their structure changes over time.

**Interface**: The term Interface is used within this document to refer to general, language-agnostic “programming interfaces”, or objects which (i) enable polymorphism, (ii) are expected to perform or provide access to some particular function or attribute, or (iii) act as a general

container for inclusion of multiple classes, or types, of objects within a field (see the term “Decorator”).

**Map**: A map is a collection of unordered key-value pairs, allowing for retrieval, updating, and deletion of values via the input of keys.

**Method**: A method is a parameterized or unparameterized procedure or function associated with an object, which either returns an output or makes a change to system state.

**NoSQL**: Used in this document to refer to database management approaches which allow for unstructured or semistructured data as opposed to preset schemas or traditional tables.

**Object**: An object, in the context of this document, unless otherwise noted, refers to data structure which has identity, state, methods, and is an instance of a defined class or subclass.

**Pointer/Reference**: A pointer is a reference to an external object. The term reference is used interchangeably with pointer throughout the document.

**Sigmoid Curve**: A sigmoid curve refers an “S” shaped curve on a graph.

**SQL**: Acronym for Structured Query Language.

**Subclass**: A subclass is an object type which inherits methods, qualities, or values from some other object type (a parent).

**Type System**: A type system, whether applied in programming or other contexts, categorizes objects and functions based on inherent attributes and structures. It dictates how these categories can be manipulated and represented, and how they interact, transform, and become realized within a given system or framework.

# Diagram Legend

**Pointer.**

Symbol: *

Example: *Object

**Array.**

Symbol: [ ]

Example: [ ] Object

**Method.**

Symbol: ( )

Example: Procedure( ):

**Map.**

Symbol: Map[ ]

Example: Map [ Key ] Value

**Subclass/Interface.**

Symbol: << >>

Example: << Parent >> SubClass

**Relationships**

All directed relationships are projected from Class A to Class B.

- **Solid Line | Black Arrow**
  - Class B contains references to Class A
- **Solid Line | No Arrow**
  - Class B is implemented within Class A
- **Dotted Line | Black Arrow**
  - Class B is a subclass or interface of, or otherwise inherits from Class A.
- **Dotted Line | White Arrow**
  - Object B is realized, instantiated, or retrieved by a method of Object A.

# Theoretical Foundations

ATLAS makes use of a ***Pattern Language*** approach, in which any identifiable object may be labeled and assigned attributes in common with other like-objects (i.e. patterns), structuring abstract, esoteric, and complicated information within niche contexts and allowing communication to occur through common reference to patterns [3]. Where traditional Pattern Language approaches are ad hoc or tailored to a particular domain (e.g., architecture [3], software [9], user experience design [10]), the version of ATLAS presented in this document facilitates a generalization and scaling of the Pattern Language approach by (i) treating all entities of interest (e.g., patterns, objects) as potential patterns or instantiations of patterns, (ii) providing a flexible container for attributes as opposed to a preset schema, (iii) treating attributes themselves as entities of interest, and (iv) incorporating a ***Question Oriented Design*** that exploits the following principles (summarized inFigure 7):

1. ***A Request for Information Contains Information***. By virtue of being self-generated, questions reveal information about an agent's situational awareness and goals. The content, format, scope and even order of queries can reveal an agent's intents and approach to exploring a cognitive space, and information about how the agent classifies the objects it presents queries about. At its core, a query is a request for information, and the specificity of the request communicates information regarding the answer *before* the answer has arrived. ATLAS treats queries which are requests for information *about* an object (i.e., the value of an attribute), as a *router* (a) between agents and some defined attribute of that object, (b) *generalizable* to other objects which also have that attribute, and (c) which may contain information regarding expectations of attributes of objects which might be values that could be held as responses to the query. This is to say that the *query* "What is the flag of this nation?" is a router (a) between a human and the attribute of some defined object representing a nation state, (b) that is generalizable to any other nation state object, and (c) that holds information regarding the answers to the query - that the object provided as a response to this query is a *flag,* with its own set of expected attributes. Thus, by establishing information quality standards (e.g., standard questions associated with like-objects, or objects which fit a particular pattern) we can both generate new requests for information and classify objects as a function of resolving requests for information. These structured queries are referred to as an *instance query* or *iQuery* in the ATLAS system.
2. ***Missing Information is Information***. One of the challenges knowledge management seeks to address is missing information, often referred to as a knowledge gap. Filling knowledge gaps is an ongoing subject of knowledge management research. The ATLAS system is built with the acknowledgement of the fact that, paradoxically, a lack of signal can communicate information. The structured query system which ATLAS implements treats missing information as a serviceable request, and cannot recognize a missing data

value which does not contain some respective routing instructions (e.g., a human legible prompt, or an API request) to serve the request to an appropriate agent. Where such a request cannot be initially handled, it reveals the missing information as a knowledge gap by default. In the case that the requests simply cannot be handled, it reveals information about the object that is the subject of the request for information, about how the request was served, or about the request for information itself. For example, it may be the case that the object is exceptional in terms of its class or pattern membership (e.g., a country which has no single national flag), or there may be an anomaly preventing resolution (e.g., a missing data value from a data set expected to have a value).

3. ***Disagreement Over Information is Information***. Conflicting information is an intrinsic part of informational landscapes. Instead of resolving the issue and negating the "wrong" answer, proposed here is an acknowledgement that discord *is* signal, and measurable. Disagreement over attribute values may be due to error, differences in scope, or due to variable interpretations of the same observations. As such, ATLAS is designed for local "instancing" of and sharing among knowledge bases as opposed to development as a single, universal knowledge base - with an intent to allow global consistency of common reference without requiring global consistency in standards, schema, or data values. So long as there are opportunities for common reference and documentation, knowledge bases become comparable, and as such, disagreement over standards, schema, and data values become measurable and resolvable without necessarily requiring disruptive rip-and-replace conformance or adoption of new standards.

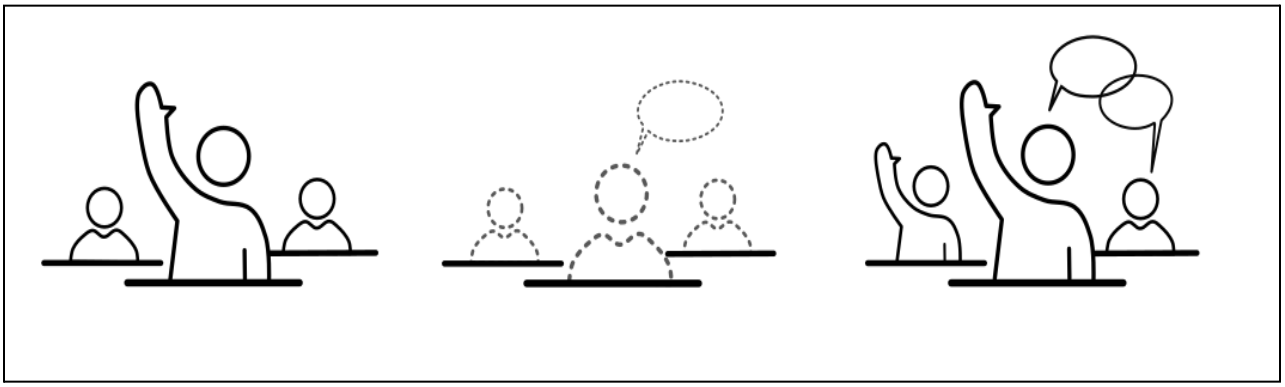

Figure 7. Three kinds of information contained in question systems. In a classroom setting, students raise hands to ask a question. We posit that (a) a question contains information, (b) lack of answers from the students contains information, (c) disagreement over answers among students contains information.

# System Definition

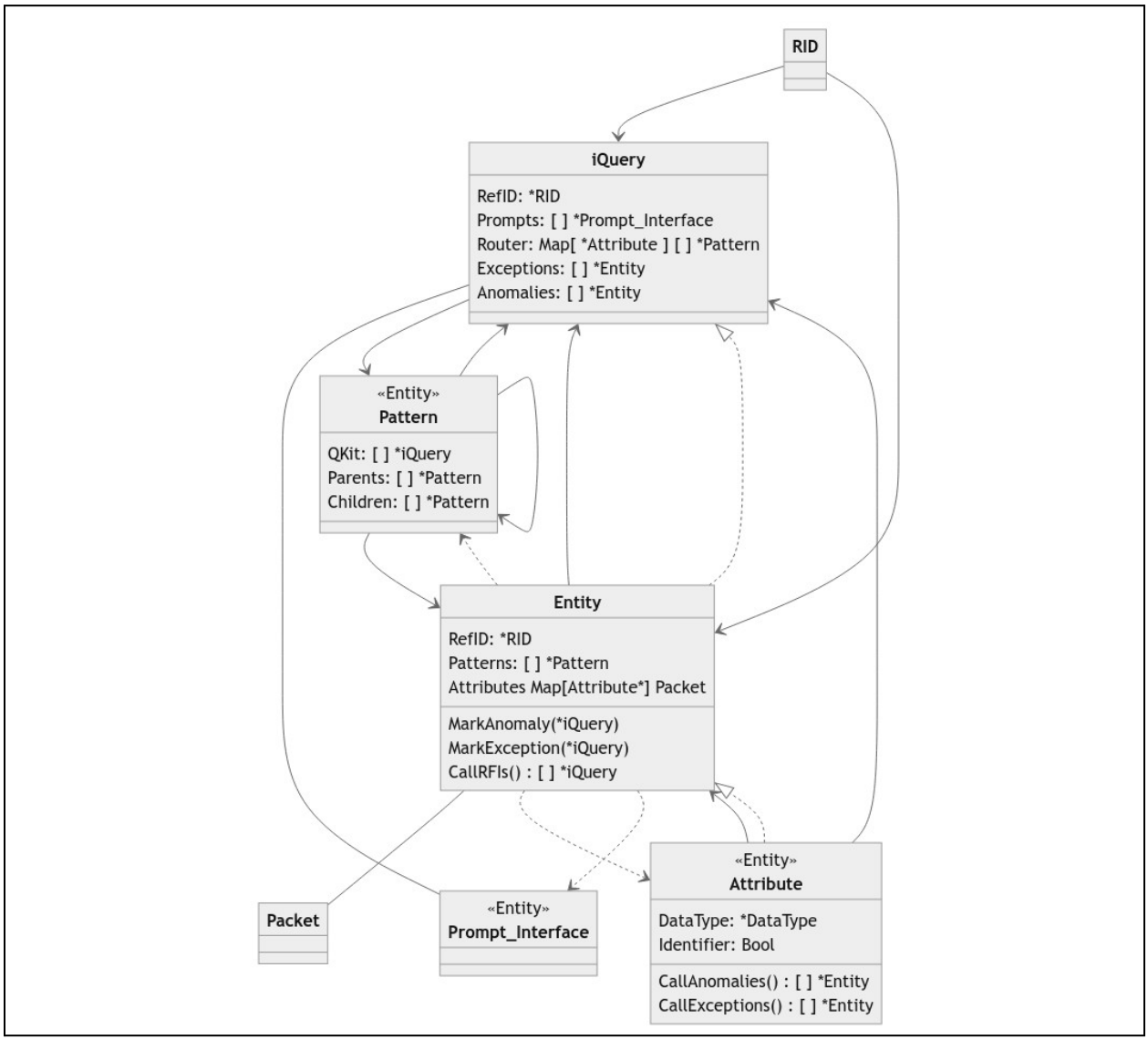


Figure 8. Representation of the full ATLAS system.

Here, each of the core components of the ATLAS system, and their respective essential properties are defined and described. The ATLAS system is intended to allow for numerous stand-alone “instances” (referred to as an ATLAS instance), containing only several vital objects. Reference IDs (RIDs), as noted in the Section regarding Scope, are not addressed in the definition, but are noted as an essential property of core ATLAS objects - because they represent an anonymous structure, they are referenced as an object as opposed to a String. Also included in the systems definition are designations for ATLAS instances capable of communicating.

# Entity

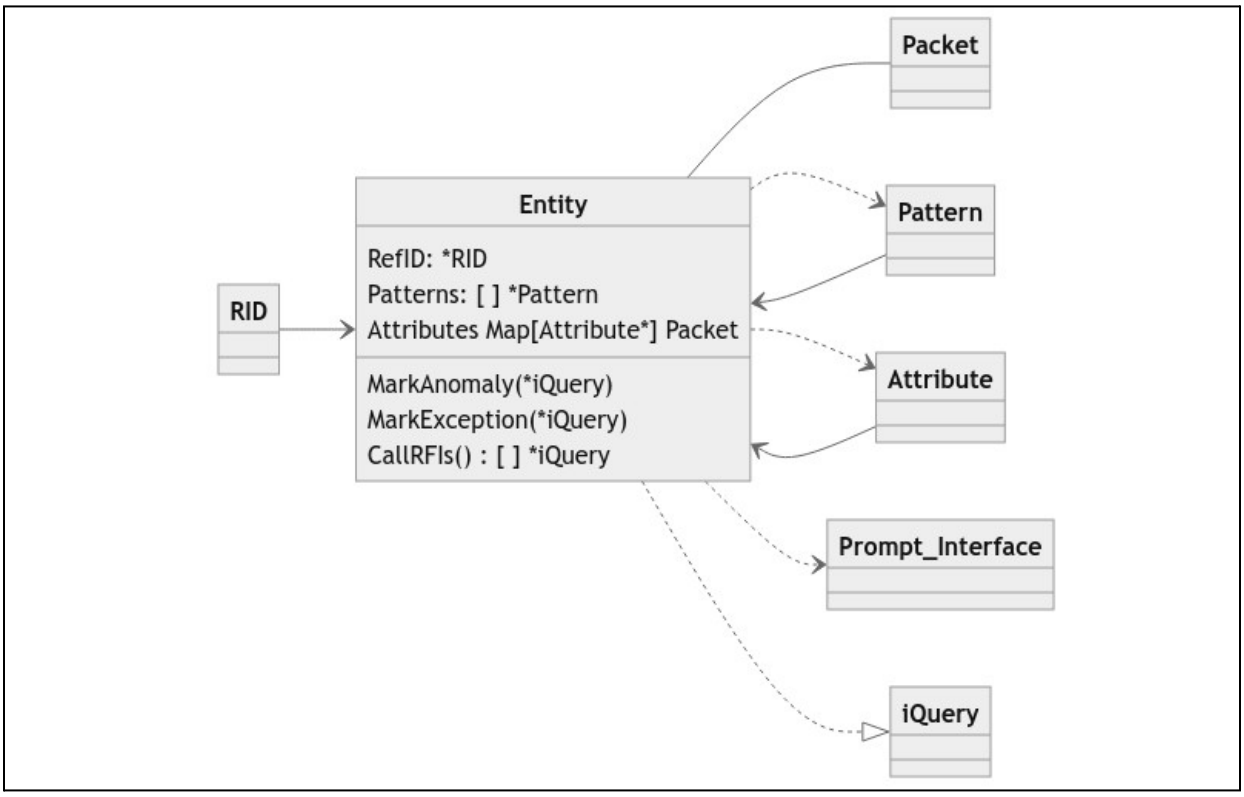


Figure 9. Entity component definition

Objects of the Entity class are used to represent and enable common reference to tangible or abstract objects and patterns of objects in physical, digital, or conceptual space. The Pattern, Attribute, and Prompt Interface classes are each subclasses which inherit from the Entity class. This class contains the following properties and methods:

**Properties**

RefID

- A pointer to a RID, used for managing merge, import, and export operations with other ATLAS entities.

Patterns

- An array of references to Pattern objects, which the Entity is an instance of.

Attributes

- A map from Attribute references to Packets, or generalized containers for data. These containers are left flexible to allow DDOs to create or integrate any data structures appropriate for their use case. The kinds of Attributes associated with an Entity are held as a retrievable array of keys to allow for flexibility in

assignment. Attributes generally become entered into an Entity attribute map via an iQuery, but an Attribute can also be manually assigned.

**Methods**

Mark Anomaly
- A procedure for marking this Entity as an anomaly to a specific iQuery (see Section titled iQuery for more information on anomalies).

Mark Exception
- A procedure for marking this Entity as an exception in relation to a specific iQuery (see Section titled iQuery for more information on exceptions).

CallRFIs
- A procedure for calling open requests for information (RFI), generated from Entity Attributes which both (i) have no values and (ii) have iQueries through which the Entity has not been marked as an anomaly or exception.

The Attributes associated with an ATLAS Entity, in NoSQL fashion, have no dedicated schema. Entity objects may contain any number of functional Attributes.

# Pattern

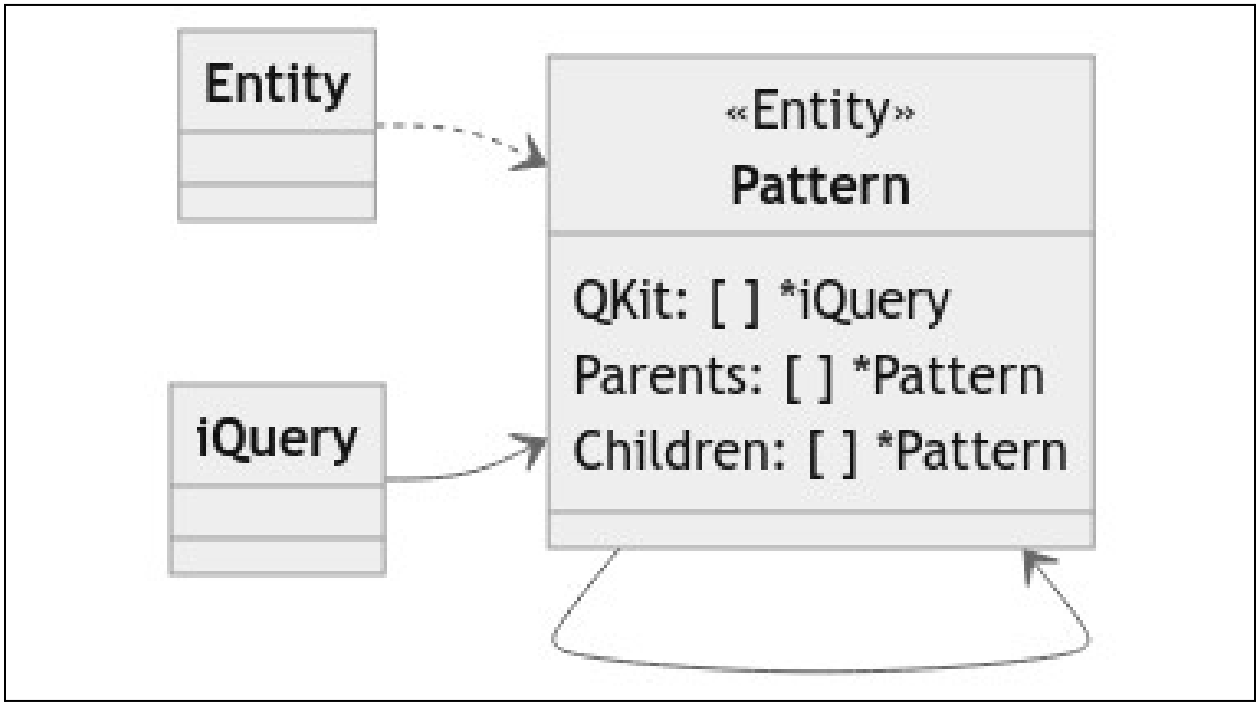


Figure 10. Pattern component definition.

Patterns are a subclass of the Entity class, representing patterns of abstract phenomena and objects which other Entity objects might be instantiations or examples of. Entity objects contain the following properties:

**Properties**

QKit

- QKit, short for *Question Kit*, contains an array of references to iQuery objects that represent the set of requests for information which are expected to be made and resolved if an object is assigned this pattern.

Parents

- The Parents array contains Patterns with the following property: any Entity assigned to this Pattern object should also be assigned to each pattern in the Parents array. For example, a Pattern “car” might have the Parent Pattern “vehicle”, meaning that any Entity given the pattern “car” would also be assigned the Pattern “vehicle”.

Children

- Children holds an array of Pattern objects to which this Pattern object is a Parent.

## iQuery

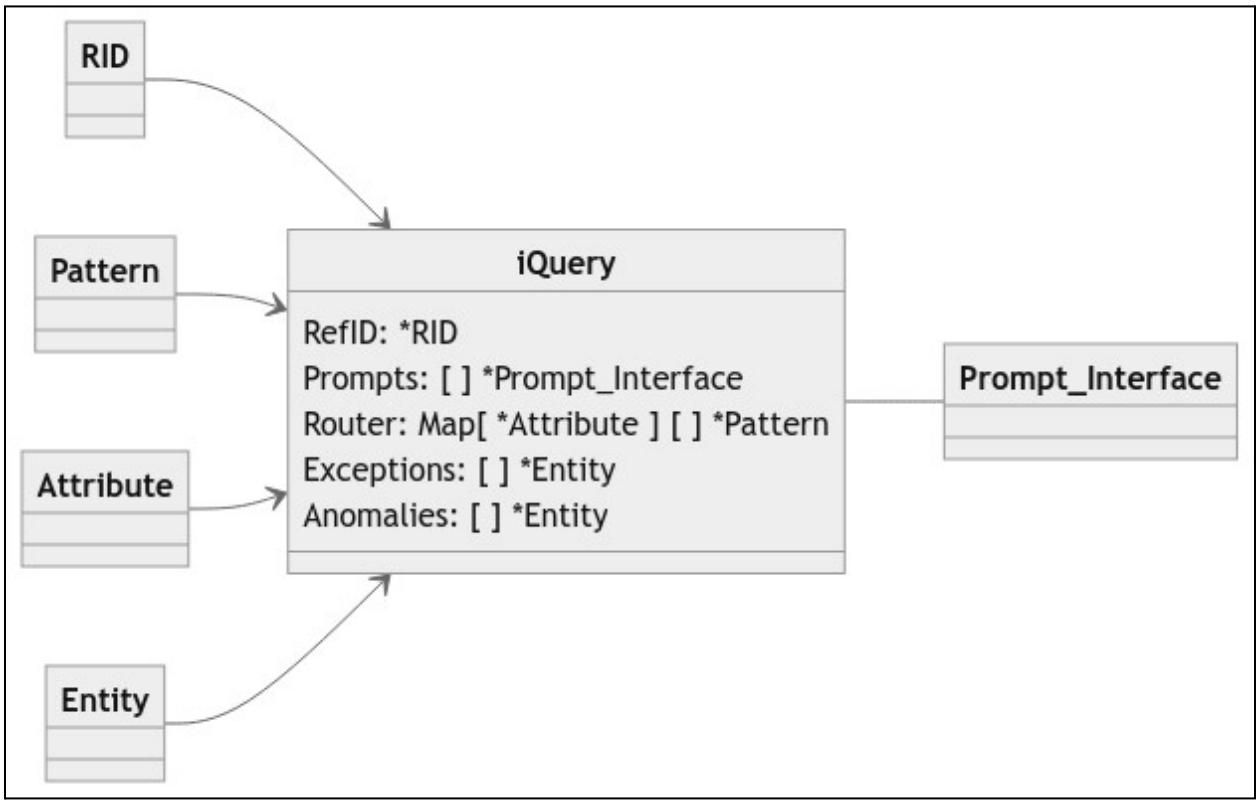


Figure 11. iQuery component definition.

The iQuery class, short for *itemized query*, are structured queries or "probes" intended to manage and facilitate the resolution of requests for information and leverage the latent information in such requests. In the ATLAS system, the iQuery functions as a router, capturing all information possible from both the resolution of the request and the implicit information in the request itself. They contain, in the form of Prompt Interfaces, an array of (i) potential approaches for requesting, receiving, and retrieving information from within and without the ATLAS, from humans, formulas, webhooks, API calls, survey instruments, algorithms, LLMs, or other computational, mathematical, social, or mixed methods and (ii) methods to package resulting information into the appropriate Attributes of the Entity subjected to the iQuery. Where the resulting value or values being routed to a particular Attribute are or include discrete ATLAS Entity objects, as opposed to some other Data Type (e.g., a String, Datetime, or Integer), the iQuery also helps to assign Patterns to those resulting Entity values. For example, where the Attribute of interest is "Original Publisher" and the human-legible prompt is "What Organization is the Original Publisher of this document?", there is information latent in the request for information - any Entity which is provided as a response to the question might fit the Patterns "Organization" or "Publisher". As such, the iQuery also functions as a generator, as Patterns contain "QKits", or arrays of iQuery objects which should be attached to any Entity assigned

said Pattern, therefore iQuery objects, through their automatic assignment of Patterns in resolution of requests for information generate new requests for information.

There are three properties that distinguish iQuery from e.g. an empty database value, a question on an online forum, or a request in a wiki editing system:

1. An iQuery is generalizable across all Entity objects of similar type, this is to say that it *can represent a simple question*, but it also allows that question to automatically be asked of every object which fits a given Pattern. For instance, if we ask “what is the Original Publisher of this document?”, this iQuery is automatically requested of every Entity which fits the Pattern “Document”, by merit of being contained within the “Document” Pattern.
2. An iQuery not only routes resulting information to an Attribute value, but also utilizes latent information to assist in the dynamic typing of defined objects (e.g. Entity objects) in order to generate new iQuery objects. In other words, each iQuery holds the potential to both (i) exponentially expand the knowledge base and (ii) manage exponential expansion through careful structure and networking of existing components.
3. An iQuery reveals knowledge gaps by default, demanding resolution of information requests within the ATLAS, as well as allowing for alternative methods to resolve requests where the requested information is simply unavailable; this feature helps avoid erroneous or impossible requests. Specifically, the iQuery allows the marking of Exceptions and Anomalies (described below). Further, because (i) any Attribute can be assigned to any Entity and (ii) an iQuery is the basis for routing requests, an Attribute with an empty data value only continues to request information if the relevant Entity has an iQuery which routes to that Attribute - which means that reassignment of Patterns allows for rapid removal of erroneous and impossible requests without impacting existing valid data.

The iQuery class is one of the few objects in the ATLAS system that does not inherit from Entity, instead acting as the core communications component among other ATLAS objects and the world outside the ATLAS instance. The iQuery class contains the following properties:

**Properties**

RefID
- A pointer to a RID used for managing merge, import, and export operations with other ATLAS entities.

Prompts
- An array of references to Prompt Interfaces, which are used to request, receive, and retrieve information and package them into Data Types which can be received by appropriate ATLAS attributes.

Router

- The Router contains a map of references to Attributes to arrays of Patterns which should be assigned to Entity objects which are or are included within data packets routed to said Attributes.

Exceptions

- The Exceptions property contains an array of Entities which, by merit of an assigned Pattern, were assigned this iQuery, but cannot or should not have a response to it, by merit of the Entity being an exceptional object. For example, consider an ATLAS Instance which includes a Pattern “US President”, which has an iQuery and an associated Attribute “date of inaugural address”. Entity objects representing US Presidents who did not perform an inaugural address, such as Tyler, Fillmore, Johnson, Arthur, or Ford, might be labeled as an exception; in the future, an analysis of exceptions might reveal the need for new Patterns which are child Patterns of US President (i.e., Presidents which gave Inaugural Addresses and Presidents which did not give Inaugural Addresses), or, in the case the ATLAS is not specifically concerned with such distinctions, such Entity objects can simply remain exceptions.

Anomalies

- The Anomalies property contains Entities which, by merit of an assigned Pattern, were assigned this iQuery, but either (i) cannot or should not have a response because the Entity it is associated with may have been assigned a pattern inappropriately, or (ii) have related automated or default Prompt Interfaces which have generated data value collision indicating (a) reliability issues in data sourcing, (b) errors in Prompt Interfaces, or (c) potential need to split or restructure Attributes.

## Attribute

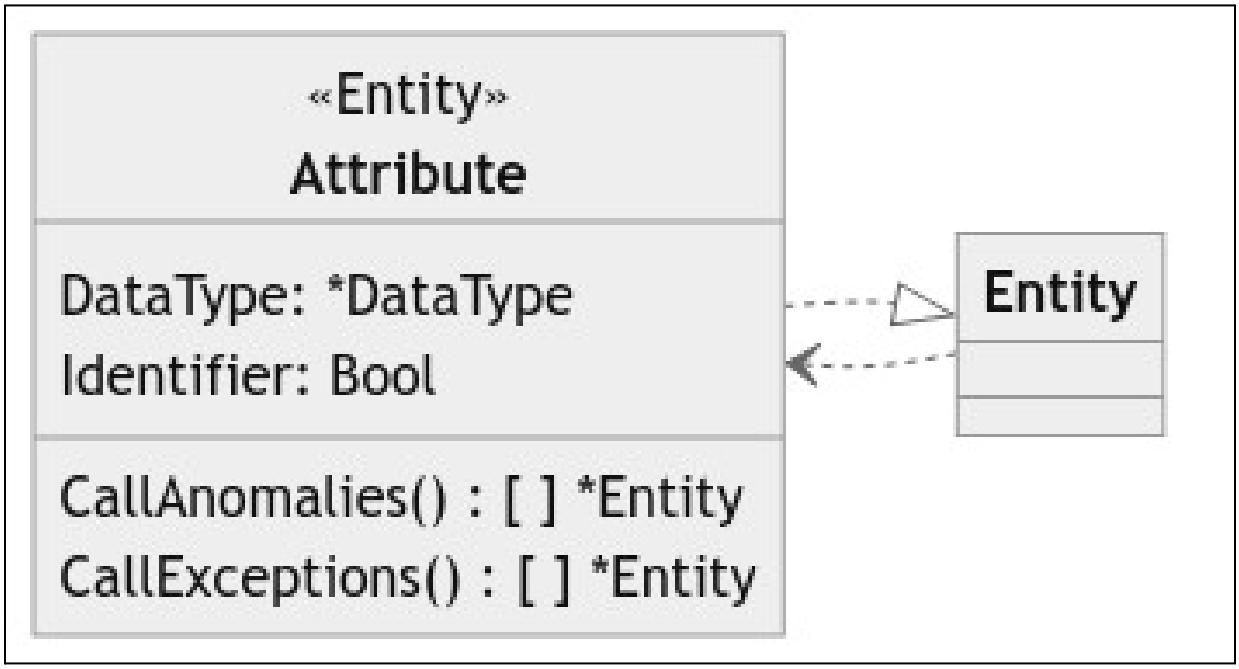


Figure 12. Attribute component definition.

The Attribute class is a subclass of Entity, used to represent any Entity which might also represent a property of another Entity. For example, “Date of Publication”, “Publisher”, and “Title” could all constitute Attribute objects. In order to enable their usage in the ATLAS system, they extend the Entity class to include the following properties and methods:

**Properties**

DataType

- Data Type contains a reference to a either a primitive (e.g., integer, float, String) or a DDO generated data structure, informing the ATLAS Instance of what kind of information structure is placed in an Entity’s Attribute map where this Attribute is listed. This is left very flexible, to ensure that any data structure the DDO sees as appropriate may be used.

Identifier

- Identifier contains a Boolean, which, if True, indicates that this Attribute may be used as an “identifier” of an Entity, for example a name, picture, ID, or other identifying value.

**Methods**

** These Methods are not necessary for an ATLAS to function, but were included in response to feedback in order to express how an ATLAS itself can serve as data for analytics that can be implemented by a DDO.*

Call Anomalies

- Call Anomalies is a procedure which queries the ATLAS to retrieve all Entities that (i) had iQuery objects linked to this Attribute, and (ii) were marked as anomalies, helping to analyze reliability of data import pipelines enabled by Prompt Interfaces.

Call Exceptions

- Call Exceptions is a procedure which queries the ATLAS to retrieve all Entities that (i) had iQuery objects linked to this Attribute, and (ii) were marked as exceptions, helping to analyze quality and stability of and patterns within the Pattern structure implemented in the ATLAS (e.g., helping to discover Pattern objects with many shared Attributes, or Pattern objects with excessive exceptions).

# Prompt Interface

Prompt Interface objects contain (i) potential approaches to requesting, receiving, and retrieving information from within and without the ATLAS, from humans or formulas, webhooks, API calls, survey instruments, algorithms, LLMs, or other computational, mathematical, social, or mixed methods and (ii) methods to package resulting information into the appropriate Attributes of the Entity subjected to the iQuery objects in which they are referenced. They extend the Entity class to include technical properties or methods defined by a DDO based on what is appropriate for their use-case, limitations, and requirements. Prompt Interfaces may be as complicated or as simple as needed and can be implemented to interact with any system or subsystem capable of taking requests and serving outputs, including other ATLAS instances (see Figure 13) or the ATLAS instance itself. For example, a DDO might implement Prompt Interfaces with simple conditional statements to enable other forms of dynamic typing, or run statistics on public sentiment using APIs in relation to Entities intended to represent psychographics. The Prompt Interface class was made to extend the Entity class in order to allow ATLAS Instances to map certain aspects of their own function (e.g., Patterns of Prompt Interfaces).

## Information Exchange Environments and Verified Information Exchange Environments

The structure provided by ATLAS, as noted elsewhere, is intended to facilitate information exchange with the outside, non-ATLAS-structured world of interactions. Here the "non-ATLAS-structured world" refers to those information exchange interactions that do not make use of the ATLAS-powered IXEs or VIEs to de-risk or leverage their interactions. Information Exchange Environments and Verified Information Exchange Environment designations are intended to clarify the roles and interactions of ATLAS instances communicating with one another, and with the non-ATLAS-structured world.

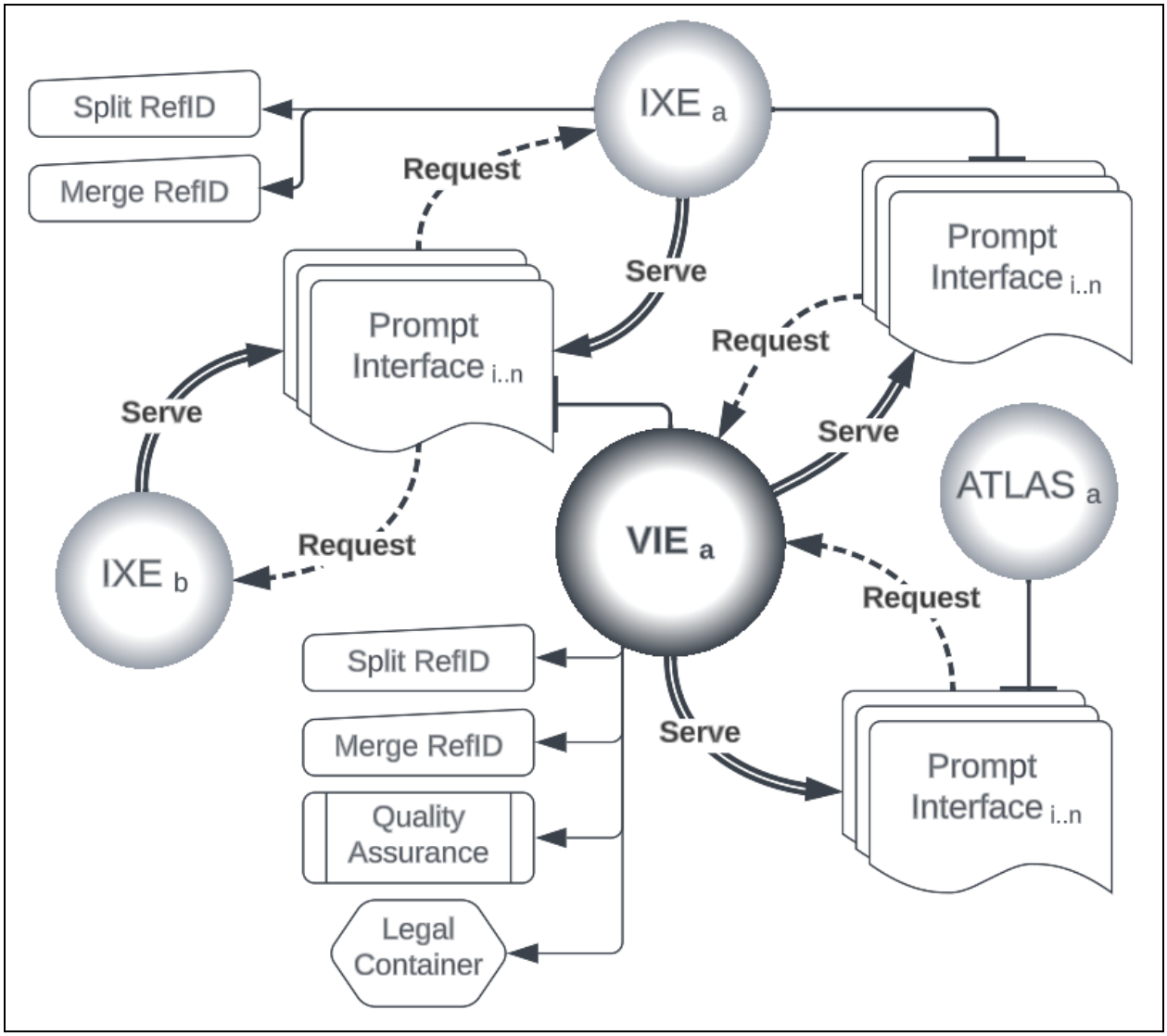


Figure 13. Schematic of interactions among Information Exchange Environments (IXE) and Verified Information Exchange Environments (VIE).

**Information Exchange Environment (IXE)**

Any ATLAS instance constitutes an Information Exchange Environment if its DDO:

1. makes some portion of the ATLAS instance's Entity and/or iQuery objects available for continuous or on-demand retrieval by another ATLAS instance's Prompt Interfaces, and,
2. provides mechanisms for (a) splitting and (b) merging of Entity and iQuery objects if the ATLAS instance itself implements continuous or on-demand retrieval from some other Information Exchange Environment.

Self-Designation as an IXE indicates to other systems that the ATLAS instance is designed, developed, and operated with anticipation of supporting exchange of iQuery objects. This designation also communicates that, if that ATLAS instance both receives and serves data, that it has the necessary mechanisms to integrate external content with internal content at scales and/or levels of complexity beyond capacity of both cognition and manual operations for individual users.

**Verified Information Exchange Environment (VIE)**

Any ATLAS instance which, via a DDO:

1. meets the criteria to constitute an Information Exchange Environment,
2. implements defined and clearly communicated quality assurance standards and procedures, as well as assigning specific duties of care to parties performing information and data management related functions and services (together "Verified Execution"), and
3. provides (i) measurable functional reliability in information availability and specifications informing expectations of structure and quality and (ii) enforcement and remedies of 3.(i) (together "Verified Enforcement").

VIEs are IXEs that offer one or more "verification" features in addition to IXE functionality. Such verification features enhance the reliability and value of VIEs for relying parties. The set of VIEs are a subset of IXEs that are organized and operated with affordances (listed in 2 and 3 above) that are intended to inform realistic organization and user expectations and to cultivate trust in specific contexts and use cases beyond the baseline functionality of an IXE. A VIE is intended to constitute a *reliable party* in a larger information supply chain or network in which it is situated. Moreover a VIE is organized and operated to provide Verified Execution and operations in accordance with explicit standards and Verified Enforcement of such standards including conflict resolution and change management capabilities such as handling issues in DMCA takedown notices, removal of illegal material, and notices of key changes and retractions. The provision of such additional "verification" affordances is not a guarantee, but reduces risk (and enhances value) for system users that rely on the system in specific and measurable ways.

# Core Mechanisms and Implications

Discussed below, are implications and mechanisms which emerge from interactions among components within the ATLAS system.

### Dynamic Typing and Management of Exponential Expansion

Given that (i) dynamic Pattern assignment occurs as a result of resolution of requests for information, (ii) new Entity objects can be generated in the process of resolving requests for information, and (iii) Pattern assignment results in new requests for information, an ATLAS instance facilitates rapid expansion of a knowledge base (see Figure 5). However, this expansion is not exponential - given that (i) the system is driven by Entity reference and structured, itemized queries (iQuery objects), (ii) resolving requests for information networks existing Entity objects, and (iii) ATLAS instances do not need to be concerned with containing all available information or patterns, only those which are relevant to a particular use case, the accumulation of new objects instead is expected to follow a sigmoid curve, wherein, the rate of accumulation will hit a key inflection point as iQuery objects begin to network more existing objects than they generate.

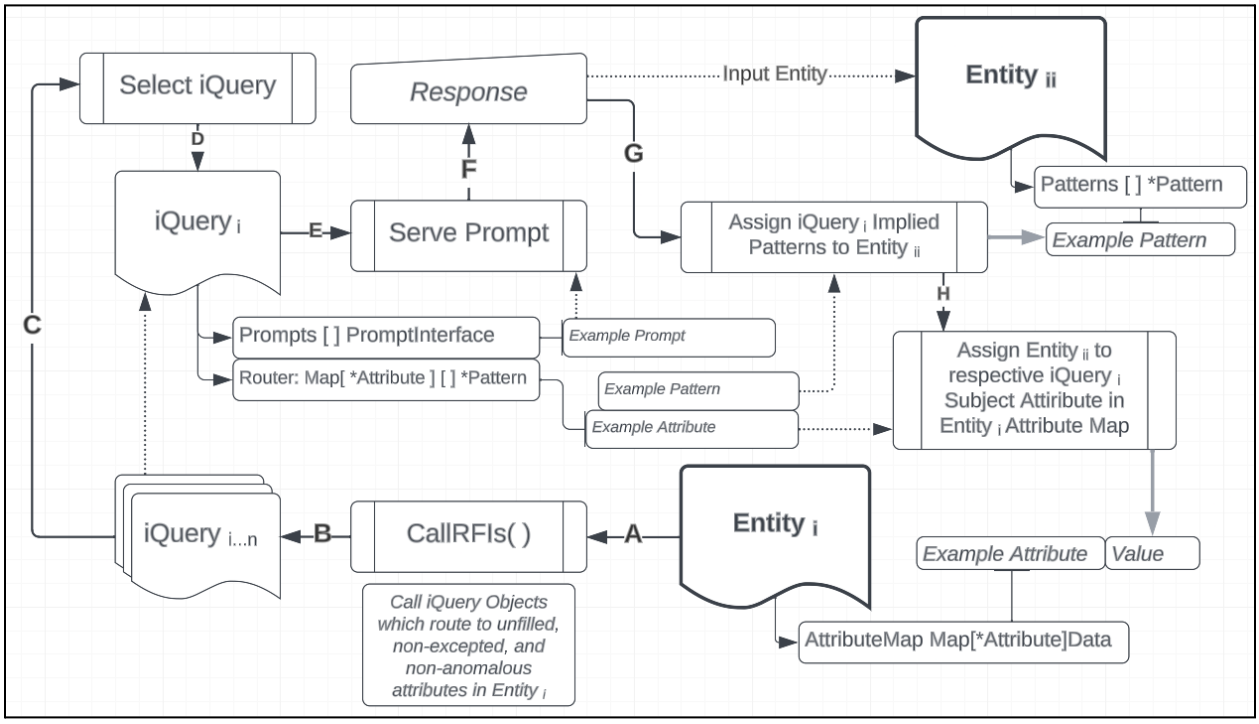


Figure 14. Representation of the role of iQuery objects in dynamic typing of Entity objects and routing of values resolving requests for information to Attributes.

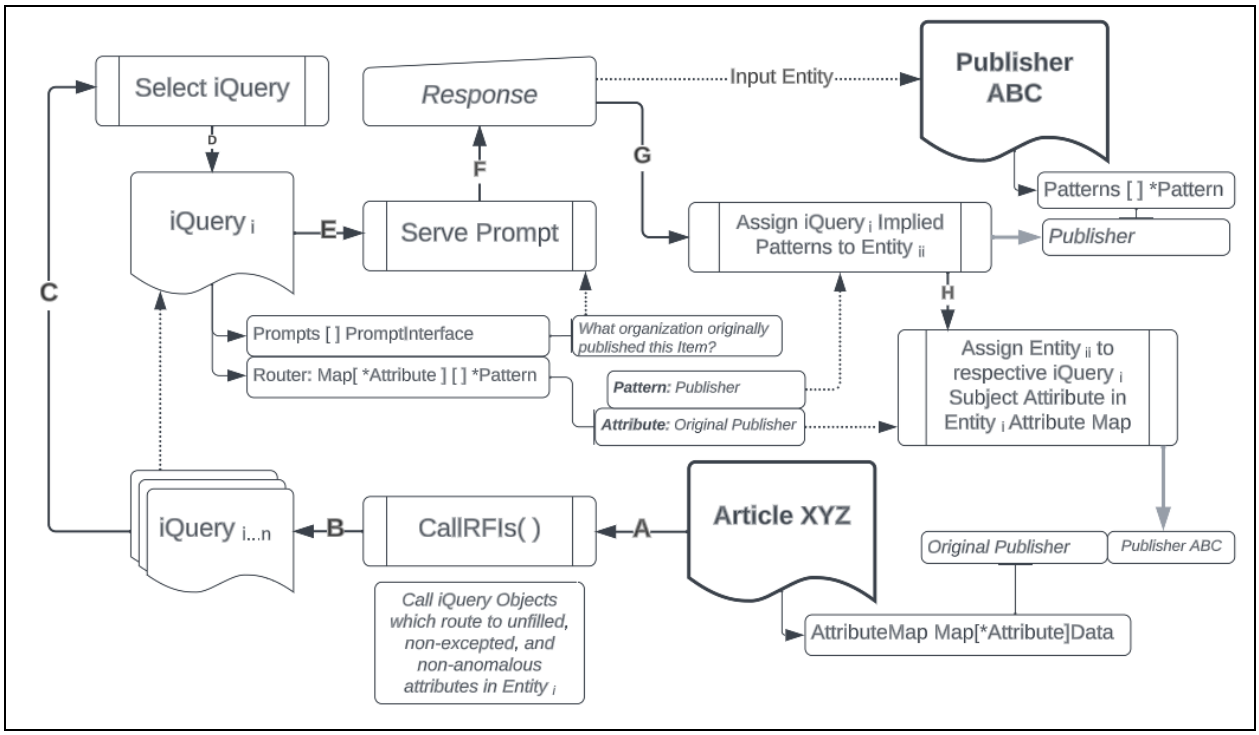


Figure 15. Representation of the role of iQuery objects using Entity objects representing an Article and a Publisher as an example.

## Information Supply Chains

Supply chains require reliability, either through quality assurance or availability of remedy. The designation of ATLAS instances as Verified Information Environments (VIEs) where responsible parties set expectations of quality assurance, functional reliability of availability and data structure, and duties of care, in combination with the flexibility of Prompt Interfaces, allows for VIEs to constitute components and *reliable parties* within a navigable information supply chain. Prompt Interface interoperability with APIs and other forms of structured data exchange could further allow for compensation in the process. It is contemplated that a variety of VIEs will be established to serve the needs of information system relying parties in a variety of contexts and domains, including governmental, commercial and proprietary, academic, and cultural. Some VIEs may offer execution and enforcement functions that will constitute “fiduciary” type intermediation, while others will not. Some VIEs will be proprietary/paywalled systems, while others will be free and open access. In each case, successful VIEs will be those that are responsive to the needs of specific populations of users in a variety of business, operating, legal, technical, and social groupings. Overall, the additional value offered by VIEs will create a VIE “layer” of the ecosystem, will offer greater regularity and choice to users in their information exchanges powered by a centralized and dynamic resource of the shared ATLAS infrastructure.

## Data Interoperability without Shared Standards

Attributes within the ATLAS system are an extension of the Entity class, which means that they carry Reference IDs and their own Attributes and Patterns, which may include documentation, version history, quality standards, and examples. Attributes can be shared between systems and linked via common Reference IDs despite differing ontology or split despite overlaps in ontology, and Prompt Interfaces can be used to call and transform data between such systems (e.g., where one system might use “datetime”, another might just use a “date” for an Attribute of the same name - or both systems might use the same name for notably different Attribute objects). Pattern assignment within Entity objects can help reveal duplicates or opportunities for transformation, or support comparability of analysis of Entity objects of varying types without requiring new Attributes. For example, “Date of Publication”, “Date of Announcement’, and “Date of Product Launch” might all share a common Pattern of “Release Date”, allowing for Entity objects which do not share each of these Attributes to be compared without manual data transformation or a new shared standard. Local ATLAS instances, by merit of Attribute objects being Entity objects, generate minimal viable documentation on their Attributes by default - allowing for interoperability of data even where schemas not agreed upon.

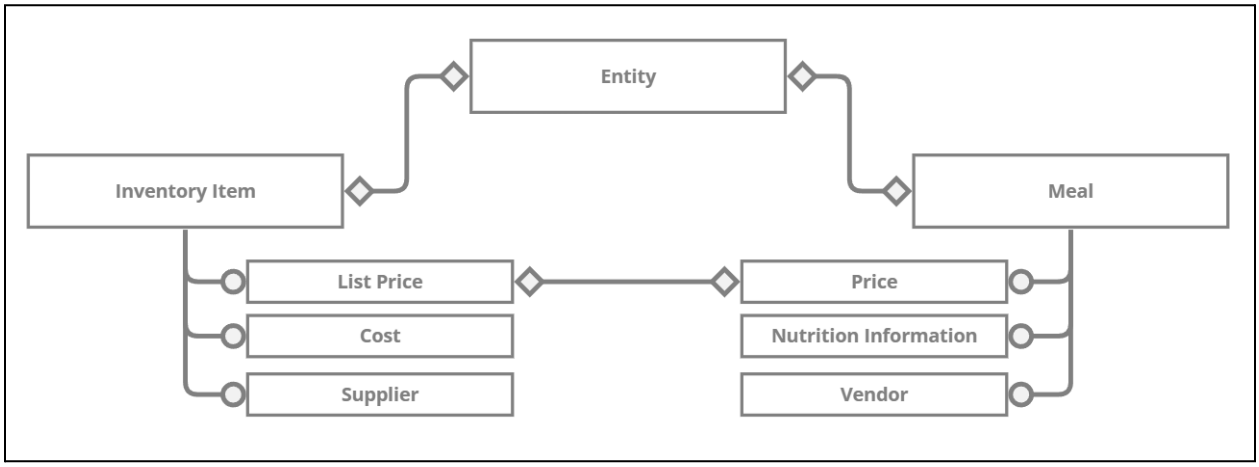


Figure 16. Representation of shared data despite varying schema and ontology.

## Restructuring and Maintaining Navigability

The nature of the components allows for several methods for handling cases where an ATLAS instance has become bloated, difficult to navigate, or where easier navigation or a new scope is necessary:

1. Filters can be used to copy an ATLAS instance and abandon the original to remove unnecessary or erroneous information without necessarily having to delete any information or damage accessibility in the future.

2. DDOs can implement version control to revert to prior snapshots and a variety of methods for duplicate detection and effective merge functions.
3. Filters and Prompt Interfaces in a new ATLAS instance can be used to request Entity objects of particular types from an existing ATLAS, in order to present a more specific version (see Figure 2) or to manage sensitive and targeted releases of information.
4. Orphan objects, or Entity objects which are not networked with the rest of the knowledge base or have no Attributes (dead objects) are immediately identifiable and easily removed if unnecessary. If the orphan Entity objects are relevant and useful to the ATLAS, but for some reason were left unconnected, manual assignment of Patterns would likely result in either navigation of previously unexplored areas of potentially relevant information, connection to the existing knowledge base, or both.
5. Any Entity can be manually assigned or unassigned Patterns and Attributes as needed, allowing for both individual edits or algorithmic (mass) edits across the entire ATLAS.

## Disagreement as Valuable Data

Shared Reference IDs resulting from exchange among supply chain components enables liquidity in sharing networked information, but also allows for disambiguation and detection of conflicts and disagreements in complicated ontologies, references, and related data. Given the ATLAS system's focus on networking local knowledge bases, as opposed to generating a single universal knowledge base, both agreement and disagreement over discrete Attribute values across communities are rendered measurable. DDOs can also implement data structures which measure such disagreement within an individual ATLAS instance as well.

## Flexible Presentation Without Changes to Underlying Data

The ATLAS system is built on the principle that there is no single authoritative view on any Entity. Patterns are used to assign collections of expected Attributes to objects, without the need to set a strict standard or schema associated with any individual Entity. As such, choices in presentation of underlying data is left flexible. Further, given that presentation is tied to discrete Attribute values, modular presentations of data with connectivity to ATLAS data can be rendered in real- or near-real time with continuous updates.

## Measurement of Knowledge Bases and Time Series Analysis

Structuring of an ATLAS instance's knowledge into discrete, referenceable components allows for measurement of disagreement over the data values of those components, as well as measurement of changes to those data values, and the accumulation of and relationships among components over time. Where ATLAS instances are deployed with iQuery objects built to interface with existing knowledge bases or LLMs, an ATLAS instance can be used as a measurement tool to reveal insights about the structure of external information sources - or, its being applied to said knowledge bases can be used as a method for measuring the quality of an

ATLAS instance's structure (e.g., iQuery and Prompt flexibility for integrating external information without human assistance).

# Future Work and Next Steps

Actionable Next Steps

- Expand and develop this document to include work in progress on topics such as hypergraph representation, modeling ATLAS as a generalized dynamical system, continuous and snapshot visualizations, use cases and adoption support, and cognitive modeling.
- Develop a pilot program to integrate ATLAS into existing knowledge management systems in targeted domains, such as healthcare, cybersecurity, or environmental science, to demonstrate its practical applications and benefits.
- Scaffold research and development focused on enhancing ATLAS's cognitive security capabilities, particularly in addressing knowledge management semantic and data integrity issues.
- Initiate collaborations with various institutions to further refine ATLAS's pattern language algorithms and question-oriented design for broader applicability.
- Investigate existing formal approaches to knowledge management systems in relevant disciplines, especially mathematics and computer science, to provide additional frameworks for describing and analyzing ATLAS structure.

Broader Future Directions

- Enhance question-oriented design in federated systems for better problem-solving and adaptability.
- Improve the usage of pattern languages for more nuanced data interpretation and context-awareness, combining insights from multiple disciplines to enrich pattern language repositories.
- Utilize modular, composable architectures to facilitate collaboration across diverse computational platforms, focusing on scalable, efficient algorithms for dynamic data handling and user query responses.
- Integrate ATLAS structure with analytical methods in statistics, logic, and cognitive modeling, for example utilizing Active Inference agents.

# References


1. Cordes RJ, Applegate-Swanson S, Friedman DA, Knight VB, Mikhailova A. Narrative Information Management. Zenodo. 2021. doi:10.5281/zenodo.5573287
2. S. David, “Atlas of Information Risk Maps”. University of Washington – Applied Physics Laboratory Information Risk Research Initiative (IRSIRI), Jul. 04, 2021. doi: 10.5281/zenodo.10292911.
3. Alexander C, Silverstein M, Ishikawa S. A Pattern Language. 1977.
4. COGSEC ATLAS Available: https://coda.io/@aien/cogsec-atlas
5. Cordes, R. J., Scott David, Ajit Maan, Alex Ruiz, Eric Sapp, Pat Scannell, and Sahil Shah. 2021. The Narrative Campaign Field Guide - First Edition. Edited by Richard J. Cordes. 1st ed. Narrative Strategies Ink.
6. Friedman DA, Cordes RJ, editors. The Great Preset: Remote Teams and Operational Art. COGSEC; 2020.
7. Blair AM. Too much to know: Managing Scholarly Information before the Modern Age. Yale University Press; 2010.
8. Cevolini A. Forgetting Machines: Knowledge Management evolution in early modern Europe. BRILL; 2016.
9. Buschmann F, Henney K, Schmidt DC. Pattern-Oriented Software Architecture, On Patterns and Pattern Languages. John Wiley & Sons; 2007.
10. UX Library - About. Available: https://www.uxlibrary.org/

# Acknowledgements

**Team**

Thank you to all those who provided feedback on development throughout 2023, and to everyone who provided last-minute feedback on the document.

**RJC**

Thanks to those who provided advisory support on information innovation behind the scenes.

Thanks to Pivot for Humanity for supporting the work.

Thank you to Michael Giangrasso, for conversations on the topic and pointing me in the right direction.