Federal Register Notice: 89 FR 51554, Federal Register :: Networking and Information Technology Research and Development Request for Information on Digital Twins Research and Development, June 18, 2024.

Request for Information on the National Digital Twins R&D Strategic Plan

DTE-FS

A Model Based System Specification for Use in Construction of an

Interoperable Digital Twin Earth Framework

DTE-FS

A Model Based System Specification for Use in Construction of an Interoperable Digital Twin Earth Framework

Version 1.0.1

Corresponding Author
[redacted], Physical Scientist, NOAA NESDIS NCEI

Contributing Authors
[redacted]v, Physical Scientist, USGS
[redacted], Software Architect, NOAA NESDIS NCEI
[redacted], Data Access Branch Chief, NOAA NESDIS NCEI

Table of Contents

Table of Contents

Abstract

Introduction

Concept Mapping

Framework Specification
- Syntactic Interoperability
- Schematic Interoperability
- Semantic Interoperability
- Legal Interoperability

Requirements Verification

Illustrative Case Study

Reference Implementation

References

Abstract

The earth science community has long been motivated in achieving harmonization of the ways it represents its domain in order to achieve a comprehensive, confident, current, and shared understanding of the earth system that is broadly available for democratic decisional use. Countless efforts across data, semantic, technology, and governance disciplines have been

undertaken toward furthering this goal, and the dream seems closer to being realized than ever. It is believed that with a proper synthesis of available efforts, a widely accessible and evolutionary digital twin earth may now be defined, constructed, and managed.

In this work we describe a specification useful in guiding construction of a model-based framework that enables implementation of a multi-viewpoint, evolutionary, and communally managed digital twin earth that is represented as a syntactically, schematically, semantically, and legally interoperable system of systems. A framework implementing this specification is capable of supporting the management of a contextually rich, user driven, readily accessible, easily manageable, and entirely flexible model of the earth system that spans its lifecycle, updates from real-time data of wide variety and source, and supports making, storing, justifying, and publishing fully traceable decisions made using machine learning, simulation, reasoning, and other process techniques.

Along with contextual grounding, this work provides a technical validation of the framework specification through description of a practical reference implementation framework - the virtual Archival Information Package (vAIP) - that is core to NOAA's upcoming Next Generation Archive and Access service. A variety of use cases of the digital twin earth framework, both in general and as implemented in the vAIP, are provided throughout this work to further refine understanding of specification configuration, utility, and implementation.

Introduction

Represented as a physical system, the earth is incomprehensibly complex, and the scope of efforts undertaken to measure it, understand it, and explain it, together with the effort-relevant context questions of who, where, when, what, why, and how, are incredibly rich in character, utility, and provenance. These humanity-spanning endeavors toward earth-system-based understandings and actions are fundamentally data-dealing, and in theory, the data they produce and consume should be readily understandable and widely reusable across various viewpoints and contexts, thanks to the tireless history of scientific effort expended in careful development and stewardship of specialized formalisms, vocabularies, lexicons, ontologies, formats, tools, and techniques that comprise the basis of holistic earth science.

However, while data related to a holistic representation of the earth system may be interoperable in theory, most implementation efforts made to take advantage of the interoperability potential toward formation of some shared system have proven to be expensive, incomplete, rigid, unsustainable, and generally unsuccessful as measured through a holistic lens. Practically unlocking the promise of universal interoperability for full data access, use, and dissemination in order to answer or better support queries through data fusion, while maintaining stable access to that interoperability through chaotic system change, has long been an end-goal of many in the earth science community, and there have been a number of approaches to its fulfillment. Some of these efforts have suited some localized domains and use cases very well under specific conditions and through particular considerations, but a general

achievement of interoperability in earth science, one that allows open access and reuse of data across disciplines, catalogs, viewpoints, and motivations, that maintains stability through evolution in character and composition, that enables natural evolution in data governance, and that scales to meet the enormous process and storage requirements of such an undertaking, has proven elusive.

While the reasons for lack of success in achieving a more universal interoperability for purposes of improving decision making based on a shared understanding of the earth system may be extremely nuanced and contextual, a reading of Conway's law, which states that "any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization's communication structure" - gives a reasonable basis of explanation. Under Conway's law, if the data produced by an organization through its systems is tightly process bound to that organization's communication structure, then the type of interoperability offered between organizations (or between internal parts of an individual organization) is limited to the types of mutually shared support in terms of both intercommunication and intracommunication.

In other words, for multiple organizations - or even multiple divisions within a given organization - to support a basic level of data exchange, they must share an understanding of syntactic interoperability; if they further want to support data integration between each other, they must share understanding of schematic interoperability; if they want to further support the ability to derive meaning from each other's data in order to enable wisdom-tier decision making, they must share an understanding of semantic interoperability; and if they want to be able to discover and reuse each others data at all, they must share understanding of legal interoperability.

This multi-faceted view of interoperability, formalized in the Group on Earth Observations (GEO) data management principles and guidelines, illustrates why general interoperability has proven to be so difficult in a scientific domain that is strongly characterized by the variety of independent efforts; at every interface boundary between organizations, defined by communication structure, there are multiple hurdles to negotiate and manage in terms of communication structure normalization between systems.

To help ground the communications issue and the value proposition of overcoming it, a quick examination of practical users and uses within the broader earth science organization produces an unsurprisingly complex panorama of scenarios. Financial institutions making investment decisions, retailers optimizing logistics, NOAA setting fisheries limits, state DEQs issuing environmental permits, USDA predicting crop yields, US Census forecasting population migrations, NASA planning satellite launches, archaeologists looking for dig sites, environmentalists coordinating pollution cleanups, utilties predicting drought, insurers providing hazard insurance, oil companies looking for drill sites, developers looking for build sites, biologists looking for new species, governments deciding on national policies, and individuals planning a camping trip, are all completely valid and significant executive-focused use cases of a physical representation of the earth system, and each represents a traditional organizational stovepipe (or collection of stovepipes) in terms of interoperable accessibility. Each also

represents its own rich set of insights, ideas, tools, techniques, and collected data that, if available for shared use, would at the least enable synthesis of a much more complete physical representation of the earth system in terms of, among other metrics, resolution, performance, and accuracy.

Under Conway's law, then, the ground truth domain complexity in terms of practical communication structures makes it clear that while earth science is fundamentally interoperable in terms of its physically described foundations, making practical use of that interoperability is limited by the practical communication-related incongruities assumed by its many implementers.The significance in overcoming these incongruities is in eliminating them as the primary limiting factor of how good any practical physical representation of the earth can be, toward an ultimate purpose of allowing all users of the representation to improve their decision making.

Based on this view, with respect to our interest, we can define our problem boundary as a set of axioms. We propose that

- **Axiom A** - Organizations and individuals (defined broadly as 'users') contribute to and study the earth system to answer questions and make decisions for reasons that are meaningful to certain other users.
- **Axiom B** - The users, questions, decisions and reasons are all infinitely variable.
- **Axiom C** - Users may greatly improve their own decision-making confidence, capacity, and capability due to improved model capability via shared representation.
- **Axiom D** - Achieving a shared representation of users is limited by the communication interfaces they share.
- **Axiom E** - Issues with communication interfaces may be defined and addressed in terms of classified interoperability.

Using these axioms, a restatement of the thesis of this work in terms of relevant context questions is possible. In these terms, our thesis may be defined as

- **What** we want to achieve (a digital representation of the physical earth that is complete, efficient, and accurate)
- For **who** we want to achieve it (users of the representation)
- The reason **why** we want to achieve it (to improve user decision making about the physical earth based on the representation)
- **When** we want to achieve it (for the length of use of the representation)
- **Where** we want to achieve it (on user infrastructure)
- **How** we can achieve it (by enabling interoperability between users)

The primary focus of this work will be in deriving and describing specification for the how, by determining requirements given defining the what, why, when, and where.

Concept Mapping

What we have defined as our primary concern may be readily understood as an attempt to support application of the concept of a ‘digital twin’ to the earth system. A **digital twin** is commonly defined as

“a virtual representation of an object or system that spans its lifecycle, is updated from real-time data, and uses simulation, machine learning and reasoning to help decision-making.” It is usually considered implicit that in this definition, ‘decision-making’ refers to decisions about improvement of the object or system itself.

The concept of a digital twin is not new, having a long history of use in industrial settings including automotives, automation, manufacturing, energy, logistics, and utilities. Any system that uses remote sensing data to represent and make decisions about itself may benefit by being modeled as a digital twin. Functionally, digital twins are primarily defined by providing a system with capabilities that include

- **Operations optimization** via simulation and monitoring
- **Predictive maintenance** recommendations based on holistic or targeted analysis of system properties
- **Anomaly detection** based on historical trend reliant models
- Fault isolation via reasoned **root cause analysis**

Applying our boundary value axioms to the basic definition of a digital twin, we might specify a **digital twin earth** (DTE) as

“a virtual, dynamic, up-to-date, physical representation of a multi-user earth system of systems that enables making decisions about the earth through the use of process techniques including simulation, machine learning, and reasoning.”

We know through Axiom D that communication structure is the defining feature of coherence in a system, and through Axiom E, that communication structures can be defined in terms of distinct interoperability focus. Our DTE may then be defined by the syntactic, schematic, semantic, and legal communications structures that it relies on. Given the existing entrenched communications structures of intended DTE users, it is not practically feasible to construct a coherent and fully user supportive system from the top down. Our DTE must therefore be supported by a set of portable communications structures that exist independently of any user system. This leads to construction of a statement on framework methodology.

- **Framework Methodology** - *In order to support a holistic DTE, we must construct and provide a portable interoperability-model framework (DTE-F) that supports all required DTE functionalities in a manner compatible with existing communication structures of the*

systems owned by all intended DTE users through implementation of a Digital Twin Earth Framework Model Specification (DTE-FS).

Framework Specification

Syntactic Interoperability

Under our methodology constraints, if we imagine our user systems as a set of independent, uniquely identifiable nodes in global DTE space, our DTE-F may be conceptually modeled as a set of bi-directional connective edges between them. If we then apply our understanding of user systems as collections of earth-related decisional processes (Axioms A and B), we can improve the correctness of the initial model by requiring DTE-FS to model connections between user processes directly rather than their parent systems. This leads to the first specification requirement:

- **DTE-FS Requirement 1**: A DTE-F implementing DTE-FS shall provide model-based, schema-standardized, machine-readable access and control interfaces to DTE processes for the purpose of enabling universal syntactic interoperability.

Stated another way, this requirement says that by holding process as a space-spanning concept, DTE-FS fundamentally supports the construction and management of process-oriented frameworks, and DTEs constructed through a DTE-F will rely on process as the fundamental representation dimension supporting each aspect of interoperability. By enforcing exposure of all control and access of processes through schema-standardized machine readable interfaces, this requirement provides the foundation for a capability that enables a general process-to-process syntactic interoperability, while simultaneously supporting the non-functional architectural goals of general scalability through distributed ownership and a flexible (i.e., asynchronous-capable) communication protocol.

Practically, we may understand this requirement as addressing the aspect of communication structure concerned with making requests and returning responses, in a way that uses a standard vocabulary and structure, so that we may construct and pass information between distributed nodes completely, asynchronously, and contextually.

To illustrate how this may look, we can refer to the MessageAPI process specification, which uses the concept of a top-level session container, similar to a **document object model** (DOM), to structure and organize requests, which in turn consist of declarative and structured records and flow conditions. Content-complete sessions are delivered to remote processors, where they are parsed, initialized, runtime processed, and returned to the original caller as a packet of records, rejections, and original request reference.

The DOM foundation of this pattern allows for **query, construction, evaluation, validation, and execution of sessions at-a-distance** by humans or machines through a standard API and also provides an easy basis for persistent storage of the workflow for full provenance and reuse.

Further, this syntactic interoperability approach is reliant on complete encapsulation of whole immutable requests and enables a concurrent processing model for achieving horizontal scalability across DTE system users. This type of concurrent processing has been described and leveraged in Communicating Sequential Processes (CSP), as well as various process calculii, including the Actor Model, to support, among other things, full linear traces of intra-process vocabulary changes as well as process-side request throttling.

Technologies covering syntactic support in this way are widely available, generally being classified as workflow management system (WMS) related tools. While not strictly required, WMS tools which implement fully declarative information model based systems that pass full instruction sets via text may be best suited for fully meeting the syntactic requirement in the long term. The Onyx Platform, TaskAPI, MessageAPI, Argo, Amazon Web Services (AWS) Step Functions, by being fully declarative and model based, provide the ability to self-describe and be constructed at-a-distance, while enabling full replay and providing static, well known bounds on their control interfaces. Systems that rely on deeply ingrained, non-model-based coded instructions and rules, such as NiFi, Metaflow, Airflow, and Flyte, may not continue to meet the syntactic interoperability requirement in the long term.

Schematic Interoperability

While our first requirement is sufficient for ensuring support of foundational syntactic interoperability across our DTE representation, it does not address how to achieve interoperability in terms of schema, and we must address this in our framework model. Toward this end, we can first refer to the definition of process as “a series or sequence of operations, tasks, and/or procedures performed on something in order to change or preserve it”.

This definition says that every one of the DTE processes that we must support are made up of smaller, purpose-oriented tasks (subprocesses), linked together in specific ways. Specifically, process tasks are classifiable in terms of their contextual process purpose - either input, identity, transformation, or output - and each is constrained in terms of allowable types of downstream connections. Taken together, these facts lead to the DTE-FS requirements two, three, four, and five.

- **DTE-FS Requirement 2**: A DTE-F implementing DTE-FS shall provide model-exposed process structure in terms of DTE subprocess tasks.
- **DTE-FS Requirement 3**: A DTE-F implementing DTE-FS shall provide model-based, schema-standardized, machine-readable access and control interfaces for DTE tasks.

- **DTE-FS Requirement 4**: A DTE-F implementing DTE-FS shall declaratively classify the purpose of every DTE process task as one of acquisition, identity, transformation (considered to include classification), or delivery.
- **DTE-FS Requirement 5**: A DTE-F implementing DTE-FS shall enforce constraints on allowable downstream DTE task connections based on type.
    - 5a. Input tasks must connect downstream to identity tasks
    - 5b. Identity tasks must connect downstream to transformation tasks or output tasks
    - 5c. Transformation tasks may connect downstream to transformation tasks or output tasks

While these specified requirements do not on their own appear to support a general schematic interoperability of processes represented within the DTE, they do lay a needed foundation. To complete the picture, we must combine them with a functional understanding of tasks. In a functional view, a task takes some schema-structured input, does some arbitrary processing, and produces some schema-structured output. In this view, for any given process task, the computational workflow that produces some output is coupled with output structure, and both must be available together for complete contextual understanding of a given process definition or execution. Additionally, in order for a machine to automatically connect a given task to one downstream, the output structure of the first task must be known to the second so that it, or parts of it, may be used to drive the workflow of the downstream task automatically. This reasoning leads to the final requirement set related to content structure.

- **DTE-FS Requirement 6**: A DTE-F implementing DTE-FS shall provide standard, machine-readable access and control interfaces for all DTE processes and tasks in terms of coupled workflow and persistence structure.
- **DTE-FS Requirement 7**: A DTE-F implementing DTE-FS shall structure all DTE processes so that a given process may be executed to completion by syntactically interoperable submission of a complete declarative, schema-standardized machine-readable map of valued fields and conditions.
- **DTE-FS Requirement 8**: A DTE-F implementing DTE-FS shall structure all DTE tasks so that a given task may be completely machine-executed toward populating a known output storage structure.

Taken together, Requirements 2 through 8 provide the ability for complete automation of process construction, process execution, task composition and validation, task execution, and provenance analysis, and this ‘single-pane-of-glass’ approach reduces schematic interoperability to a problem of providing up front values of known key-value pair fields and conditions through a uniform interface.

To illustrate this concept in practice, returning to the MessageAPI process specification, the previously described session requests provide a standard schema pattern for a request record, so that it may be composed purely in data and submitted completely for processing.

The record pattern requires provision of a flat set of fields, each needing specification of field id, type, value, required status, and optional metadata; as well as a similar flat set of conditions used for data flow routing. The record then requires each field to be containerized in some custom pattern or patterns, which is referred to as a contextual identity container, allowing conditions to make determination of inclusion. Each container is then potentially referenced in some arbitrarily nested and/or branched pattern of transformations and/or classified labels, each stage of which specifies its own output field set. All computational paths are ultimately referred to within a connection to a defined endpoint that defines what its output records are.

When a session is initialized by a processor, any construction time logic for individual tasks, including transformation context or endpoint connection, are executed to create the session context; field and condition values are then applied as a set to create and submit a request of one or more records; and endpoint connections drive processing in a lazy way, first through up front validation of task-output-to-task-input schemas, and then while during each computation iteration. At each stage of its processing, if a task fails, it is added as a rejection for return and explanation to the original caller that lists specific fields and reasons for failure.

Through this method of coupling task workflow and output in a schema-normalized way through record-based sets of fields and conditions, defined generally by DTE-FS requirements 2 through 8, schematic interoperability across processes is enabled generally for any classifiable task. Driving a process becomes a matter of valuing known fields and conditions, and driving individual tasks becomes a matter of mapping fields needed to drive a task workflow with generated and potentially persisted output of upstream tasks. Readers should note that this section, particularly in light of requirement 6, provides the described level of schematic interoperability both inter-process and intra-process.

As a final illustration of the use of schematic interoperability in the terms laid out in this section, imagine a DTE process that receives raw station temperature measurements for some given location over an hour time, does a time-interval average of them, validates the average against known seasonal expectations based on historical measurements, and then packages validated ones as a NetCDF file for reporting.

In our defined framework, this process may be modeled as consisting of five distinct tasks - An input task, which takes station measurements; a first transformation task, which does some aggregation based time-interval average; a second transformation task, which does some validation; a third transformation task, which converts file type into NetCDF format; and a final output task, which sends knowledge of the process execution to some endpoint. Each of these tasks in the process has its own functional workflow and output schema. By requirement, the process itself defines an acyclic digraph of process order, and a final output schema. In order to construct the process within the framework, the tasks must first be linked together.

Assuming a known workflow and output structure is already known for each, they can be linked together automatically, matching input requirements for a given task workflow to output from the previous task, and applying general constraints of the system in terms of classified task type. As

an example of this, the precipitation task might get raw measurements as strings, in which case there might be a workflow key called ‘precip value’, and one called ‘precip units’. This task might specify as its output structure a single JSON map of precip value and structure, called ‘precip object’.

The next task, the aggregation transformer, might take as a workflow input value a list of ‘precip objects’ and a ‘historical file location’. Since this ‘historical file location’ is not provided by the output of its upstream task, it will be required to be provided in the flat field set as a model parameter to begin execution of the overall process, and the machine can inform its caller about this fact. The NetCDF transformation task might use a ‘NetCDF template’ field, and a ‘normalized precip set’ in its workflow. Again, if the NetCDF template field isn’t provided as output from the upstream task, it will be called out as a flat-map input requirement for use at session initialization.

Careful reading of this process approach may glean that while it provides a generalized strategy for schematic interoperability, there are some shortcomings - i.e. other than key to key matching for field values, how would a machine know that one key’s content actually matches with its target? This leads us to the derivation of semantic interoperability related requirements for our DTE-FS.

Semantic Interoperability

While the DTE-FS as described thus far covers support of both generalized syntactic and schematic interoperability, it has not addressed the critical capability of semantic interoperability. Semantic interoperability between two nodes with respect to some topic is broadly defined as a shared understanding of meaning of that topic. To enable interoperability in this way, we must be able to provide the ability to understand the context of a thing, what kind of contexts a thing might belong to, reason about whether two things are the same, if they are different, if they are compatible, in what ways they are compatible, in what ways they are different, if they are related, if they are completely different, and so on.

The study of knowledge graphs has been concerned with this type of interoperability for many years, and has made great strides in its enablement, in terms of formalisms, tools, and techniques. The concepts of vocabularies, lexicons, thesaurus, knowledge organization systems, and ontologies in particular are well fleshed out systems for addressing semantic interoperability. Ontologies are patterns of understanding about some topic, using hierarchical classification of concepts and relationships between them to allow patterned storage of data and inference based on new data.

It may seem, then, that an ontology or defined vocabulary is the best way to approach providing interoperability support within the DTE-FS, and this is the approach that is taken. However, when talking about generalizing semantic interoperability between arbitrary DTE users, in order to support semantic interoperability across an enormous variety of highly specialized use cases, it also becomes obvious that a single ontology by itself is not enough - because, to support

semantic interoperability across unbounded and unknown use case, an ontology by itself would grow unbounded - thus rendering the system essentially unstable and unsustainable. So, while the concepts of ontology are indeed foundational to addressing semantic interoperability, they must be augmented with further ceremony in order to meet the requirement in letter and spirit.

To do this, we combine a number of complementary concepts to ontology. First we introduce the concept of **'fuzzy semantic interoperability'** via prototype patterned archetypes. In this approach, a bounded and flexible reference model (i.e., one that has some support for recursion and composition) that spans the desired domain is selected and converted into ontology. This reference model derived ontology then forms the foundation for allowing DTE users to build user-specific archetypes, also referred to as empty human-labeled structures. These structures are unvalued instances that derive from the small class set of the reference model ontology, providing intrinsic interoperability, but they are also human readable and easily composable, thanks to their open ended human labeling.

A quick understanding of the utility in this may be understood by this short example - if two DTE users both use the concept of a 'granule', but use their own properties to define the structure of the granule (i.e., one requires a DOI, the other requires a UUID and a DOI, one requires a checksum, the other requires two checksums, one holds the file directly, one holds a link to the file, and so on), each user may define the concept of granule in a very specific and highly customized way, but do it based on the same semantic ontology - so that either user, or another DTE user entirely, may discover both types of 'granule' together.

In order to support the requirements of the storage structures described for use in the schematic interoperability toolset, archetypes defined within the DTE-FS must support one of the distinct task types of input, identity, transformation (including classification), or output.

- **DTE-FS Requirement 9**: A DTE-F implementing DTE-FS shall provide a small, static, space-complete, and flexible classification-oriented reference model derived ontology for use in defining DTE process task produced structures.
- **DTE-FS Requirement 10**: A DTE-F implementing DTE-FS shall enforce that all DTE process tasks define their process output in terms of partially valued archetypes, known as process-specific templates, that derive directly from 'fuzzily interoperable' archetypes, also known as unvalued human labeled structures, which in turn derive from classes in the implemented DTE-F reference model ontology.
- **DTE-FS Requirement 11**: A DTE-F implementing DTE-FS shall enforce that all DTE process tasks define their output archetype in terms of the relevant supporting archetype class, either input, identity, transformation, or output.

Furthermore, as transformations likely involve field set modifications for their output, the generic archetype for transformation must support use of arbitrary ontology to contextualize identified task outputs. To accommodate this need we introduce the second concept needed to augment ontology in support of generalized semantic interoperability, which is contextualized knowledge of data. To do this, we must require that our reference model ontology support rich

contextualization of the data it holds - it must be able to describe data in terms of its relationships to potentially rich networks of structural, semantic, and other representation, it also must be able to describe data in terms of its packaging and preservation information.

- **DTE-FS Requirement 12**: A DTE-F implementing DTE-FS shall provide accommodation for process transformation tasks in supporting ontological restructuring of upstream task output within arbitrary and machine-accessible context in terms of content and character.

What this means practically is best illustrated through example. Going back to the previous section and our discussion on schematic interoperability, we have a task in our sample process that produces NetCDF files. This is a transformation task, so it must be supported by an archetype related to transformation. If the transformation output archetype has a structure of two fields, one the key label of ‘netCDF file’, and one with label of ‘netCDF template’ the archetype must include links to or direct text that describes that this is a known ontology; what URI to load the ontology namespace from; where to find the schema; potentially what fields in the netCDF file schema mean; etc. Through this approach, we might support any known ontology, within our reference model ontology, and also, through the use of modern tools like NLP, enable machine access to understanding and parsing of arbitrary context.

Another common use of this approach to semantic interoperability within the DTE is in a contextual assessment, or qualified quality control, of given identified entities. As a distributed framework supporting arbitrary users, there can be no guarantee of the quality of something that is produced, other than through the lens of contextual certification. For example, there may be 100 DTE users that produce rainfall data of varying qualities. If another user needs to use the output from each, but weight each output differently, they may first run each through an assessment transformation characterized by an evaluative metric space structure - e.g. confidence score, accuracy, precision, etc. alternatively, a model or simulation transformation may be run with and without each data in order to determine these scores and then use them in a weighted or considered way depending on context.

In any case, the result is the ability for multiple users in multiple contexts to provide a ‘goodness’ type score, in a specific metric, and make that score context available to other users for discovery and use along with the originally identified data. Imagine if NOAA or another authoritative agency were to use crowd sourced data in a product, they might certify it first through some quality score, and then other users could use semantic contextualization for search during data mining.

This approach also provides a mechanism for constructing and feeding multiple semantically discoverable and interoperable viewpoints to monitor the system health of the DTE itself, in terms of security viewpoints, integrity viewpoints, performance viewpoints, and others. Viewpoints may represent parameter sets and machine-accessible usability context to feed graphical user interface (GUI) analysis tools, multi-layered inference ontologies for inference based reasoning, and others.

There are several generic ontologies that may meet the requirements laid out in this section, including OAI-ORE, PROV-ES, OAIS-VAIP, and OpenEHR. This is not an exhaustive list of base level ontologies that may support the requirements, and there are several existing standards and ones in development that may be provided by a DTE-F to meet the needs of a given DTE. It is strongly recommended, but not required, that any DTE-F implementing DTE-FS use existing ontologies based on standard, self-contained, and space complete reference models. While a DTE-F backing ontology may be constructed ad-hoc from various reference models or on a case by case basis, this strategy may result in violation of requirement and invalidation of the DTE over time, if the ontology is found to be unbounded, space-incomplete, or otherwise incongruous with evolutionary growth of the specified domain.

Legal Interoperability

The last interoperability toolset targeted for support by the DTE-FS framework is that of legal interoperability, which in our context deals with whether and/or what level of access a given DTE user process has to another DTE user process, its tasks, and/or its data. This is particularly important to address in the holistic DTE that is envisioned for support, as not every user of the system might want to make its processes and process related data available to every other user in the same ways. There are two aspects of legal interoperability control that are considered by DTE-FS - first, the ability to discover rights, and second, the ability to enforce access rights.

The ability to discover and assess access rights, including the method of requesting accommodation via those access rights, should be handled through use of archetype augmentation within the semantic framework, and thus we have a new requirement for the semantic framework to be able to handle management of attached access rights information on both processes themselves and all of their content. This seemingly small modification to our reference model requirements has rather broad implications for our system, as it now places an additional constraint on the choice of reference model, as well as a less obvious constraint on processes themselves - as with this additional requirement, processes can now be inferred to need their own archetype that fits within the reference model.

- **DTE-FS Requirement 13**: A DTE-F implementing DTE-FS shall require that all DTE assets described by an archetype-derived storage structure include easily accessible access rights.
- **DTE-FS Requirement 14**: A DTE-F implementing DTE-FS shall require that DTE processes be described through an archetype derived from the implemented DTE-F reference model ontology that includes access rights information.
- **DTE-FS Requirement 15**: A DTE-F implementing DTE-FS shall require that DTE processes described by an archetype derived template be provided standard, machine-readable access and control interfaces.

The access rights that describe the asset they are attached to should provide a machine-readable representation of the specific access rights policy, in terms of whether or not

a particular hopeful accessor can retrieve them, and then if allowed, provide machine-readable instructions to what ways the accessor should go about requesting the retrieval. This may involve, for example, a link to a token-request process that takes a user id and some password, generates a time-sensitive token, and then enables this token for use in running the process.

- **DTE-FS Requirement 16**: A DTE-F implementing DTE-FS shall require that access rights attached to a given DTE asset provide machine-readable capability for automatically using assets in the ways they are allowed.

Requirements Verification

With the basic framework derivation complete, it is important to first return to the DTE functional definitions to assess whether or not all of the functionality that defines a DTE is supported by the specified requirements model. We do this now by first describing the functional requirement, then assessing if and how it is supported in turn, adding new DTE-FS requirements as needed.

- **Functional Support Requirement:** Operations optimization via simulation and monitoring.
    - Simulation usually involves running a known model many times with ensembles of input data, according to some distribution of one or more model parameters, in order to determine a range of outputs. Important for simulation are the ability to trigger the same process model with different input; the ability to store output alongside given input conditions; the ability to deliver model output to multiple visualization tools; and the ability to deploy a new version of a model for simultaneous testing in the case that operations is changed as a result of simulation.
    - Monitoring usually involves analysis of many user-consumable insights that live fairly close access to raw data. Important for monitoring support are the ability to handle data availability in real-time; the ability to transform data for use by analysis and visualization tools; and the ability to send alerts in the case of issue.

Based on these descriptions, our specification may be missing requirements related to guarantee of streaming data acceptance, support for online models, versioning of models and associated process and task control context, and the ability to feed visual and other analysis tools within the model.

To guarantee streaming data support, we should add more stringent constraints to our existing requirements of asynchronous data passing to ensure that DTE users are always available to take requests and serve responses to other users of the DTE, and in the case of brief unavailability, must recover and catch up quickly in processing requests and serving responses exactly.

- **DTE-FS Requirement 17**: A DTE-F implementing DTE-FS shall require that all DTE users accept and process all syntactically complete and legally acceptable process requests in a timely way.
- **DTE-FS Requirement 18**: A DTE-F implementing DTE-FS shall require that all DTE users immediately return a response to requesting users that contains tracking information about the request and status of the request.
- **DTE-FS Requirement 19**: A DTE-F implementing DTE-FS shall require that all DTE users recover quickly in the case of brief unavailability.

To guarantee support for online models, we should add a requirement that online models be declared as such within their task context and that online models persist information about state change.

- **DTE-FS Requirement 20**: A DTE-F implementing DTE-FS shall require that all DTE process tasks containing malleable models declare the model as malleable as part of the archetype based task context container derived from the DTE-F reference model onlology.
- **DTE-FS Requirement 21**: A DTE-F implementing DTE-FS shall require that all DTE process tasks containing mutable models persist any state changes made to them within an archetype derived task context in a way that makes them completely machine recoverable.
- **DTE-FS Requirement 22**: A DTE-F implementing DTE-FS shall require that all DTE process tasks containing mutable models make any previous state changes discoverable, accessible, and completely recoverable to any legally authorized user of the DTE.

To guarantee support for versioning of models and their context control structures, we should require that all task and process deployments persist and provide ready access to their state history.

- **DTE-FS Requirement 23**: A DTE-F implementing DTE-FS shall require that all DTE processes and DTE process tasks persist all deployments as versions within an archetype derived task context in a way that makes them completely machine recoverable.
- **DTE-FS Requirement 24**: A DTE-F implementing DTE-FS shall require that all DTE processes and DTE process tasks make all versions of themselves discoverable, accessible, and completely recoverable to any legally authorized user of the DTE.

To guarantee interoperability and support for visual tools, opinionated catalogs, and other analytical tools, we should require that visual tools and other DTE-aiding endpoints be represented through archetype by the DTE-F provided reference model derived ontology.

- **DTE-FS Requirement 25**: A DTE-F implementing DTE-FS shall require that all DTE and DTE-adjacent tools related to or assisting in DTE decisions, including visual tools,

access tools, opinionated catalogs, and other analytical tools, be persisted as DTE-F archetype structured entities.
- **DTE-FS Requirement 26**: A DTE-F implementing DTE-FS shall require that all DTE-persisted archetype structured entities be complete and available for complete machine configuration.
- **DTE-FS Requirement 27**: A DTE-F implementing DTE-FS shall require that all DTE-persisted archetype structured entities be machine findable, accessible, and controllable in all ways they are used.

- **Functional Support Requirement:** Predictive maintenance recommendations based on holistic or targeted analysis of system properties
    - In an earth system, properties may be user-specific parameters that include things like pollution levels, water levels, drought indeces, and others. Holistic analysis requires synthesis of various sources, which in turn requires identification and aggregate transformation. Targeted analysis may or may not require synthesis or splitting data apart, also requiring identification and split or join transformation. Both analyses rely on historical information and searching across data persistence. Predictive maintenance recommendations based on either of these analyses requires further transformation and delivery of results and recommendations.

Existing DTE-FS requirements cover basic machine-readable semantically interoperable information discovery, synthesis and splitting via transformation. Existing requirements also cover preservation of historical information about transformation models via versioning. However, new requirements must be added in order to cover historical analysis completely, as well as to ensure delivery of machine-readable and semantically interoperable results and recommendations.

- **DTE-FS Requirement 28**: A DTE-F implementing DTE-FS shall require that all system properties of interest be persisted as DTE-F archetype structured identified entities so they may be machine or human discovered, accessed, and contextually linked to other DTE entities through DTE processes.
- **DTE-FS Requirement 29**: A DTE-F implementing DTE-FS shall require that all DTE-produced archetype structured process output that relates to any system properties of interest be persisted and contextually linked to the system property of interest in a machine discoverable and accessible way.

- **Functional Support Requirement:** Anomaly detection based on historical trend reliant models
    - Modern anomaly detection generally relies on the use of machine learning transformation models to assess new real-time data against historical trend and expectation data. The model may require online updates through the integration of new data into the model. When anomalies are detected, notice may need to be sent as alerts to one or more recipients.

Existing DTE-FS requirements cover machine interoperable model contextualization, online model handling, historical versioning and version recovery, historical persistence of output data for retrieval, and version replay. Alerting behaviors are notionally covered, however existing requirements may be augmented to support complete knowledge of alert targets.

- **DTE-FS Requirement 30**: A DTE-F implementing DTE-FS shall require that all targets of DTE outputs for use in decisions be constructed and persisted as archetype structured identity entities so that they may be machine or human discovered, accessed, and contextually linked to other DTE entities through DTE processes.

- **Functional Support Requirement**: Fault isolation via reasoned root cause analysis
    - Root cause analysis requires full provenance tracing in terms of process workflow. In an earth system context, fault isolation might deal with attempting to determine the reason why some parameter or metric was affecting the earth in some way, which requires holistic system searching, process synthesis, and semantic inference.

Existing DTE-FS requirements cover most requirements for fault isolation capability, including historical tracking of process and task, contextualized semantic discovery for inference, and process synthesis tracking. Requirements may be augmented to more specifically support holistic system searching.

- **DTE-FS Requirement 31**: A DTE-F implementing DTE-FS shall require that all DTE archetypes be readily human or machine discoverable, accessible, and usable in a fast and efficient way.
- **DTE-FS Requirement 32**: A DTE-F implementing DTE-FS shall require that all DTE templates, derived from DTE archetypes, be readily human or machine discoverable, accessible, and usable in a fast and efficient way.
- **DTE-FS Requirement 33**: A DTE-F implementing DTE-FS shall require that all DTE entity individuals, derived from DTE templates, be readily human or machine discoverable, accessible, and usable in a fast and efficient way.

Practically, this may mean that federated archetypes, templates, and entity individuals be hierarchicaly catalogued by a DTE user, potentially in many layers, within their own archetype derived template or entity individual so that search may quickly find and drill down areas of interest in a concurrent and parallel way.

With these final derivations, the DTE functional requirements are verified, and the basic DTE-FS requirements set is functionally complete. However, it is understood that practically meeting so many requirements may be difficult during implementation, and so we introduce to DTE-FS two additional requirements, coherent with the core set, that serve to guide architectural and design implementation.

- **DTE-FS Requirement 34**: A DTE-F implementing DTE-FS shall require that all DTE users implement a small, static, DTE-F interoperability-model-complete and fully DTE-FS requirement-compatible API, in whatever language or technology required for that user.
- **DTE-FS Requirement 35**: A DTE-F implementing DTE-FS shall require that all DTE users interact with the DTE solely through the user-specifed and implemented API, building any additional interactive tools on top of the API itself.

By enforcing this API requirement, DTE-F will assure that users enable better understanding of the data model and compliance with the specified requirements in a sustainable way.

Illustrative Case Study

To better illustrate the goals, use, and utility of what we are proposing, it is useful to walk through practical examples. Toward this end, we may first consider the Virginia Department of Mines, Minerals, and Energy (DMME) as a user of our DTE. Through Surface Mining Control and Reclamation Act (SMCRA), Clean Air Act (CAA), and Clean Water Act (CWA) authorities, with oversight by the EPA and OSM, DMME owns a system (process set) responsible for making decisions about whether or not to issue, revoke, or modify energy permits, including natural gas and coal mining permits, in the state of Virginia.

Permitting decisions made by DMME are composites of multiple smaller analyses and predictions (tasks) about water quality, benthic health, hydrologic impacts to wetlands, and land use management, among others. Each of these individual analyses may be based on any number of complex and highly customized algorithms reliant on synthesis of voluminous and diverse real-time and historical experimental data. Much of this data is collected by DMME directly, by hand or automated sensor, stored and managed within a database according to some custom structure.

In the water quality process alone, there are multiple instream, non-point, groundwater, precipitation, cumulative hydrologic, and total maximum analyses across dozens of parameters done based on data from multiple real-time input data flows. The fundamental output of these analyses, a singular decision on whether or not to allow or enable some wide-scale change to the earth, is determined by the quality of the analyses which are ultimately dependent on the models and raw data they have access to. Improving the decision, therefore, is reliant on improving the analyses, in turn reliant on improving the models, in turn reliant on the availability and quality of raw data. Discovery, construction, and/or management of these decisions and associated analyses, models, and data is also heavily dependent on the subject matter experts (SMEs) in staying abreast of current implementation and new developments; this is generally a slow, expensive, and uncontrolled process.

Let us now consider a second user of our DTE, NOAA, which may be thought of as a large and multi-system (process) owning DTE user. One of the systems (processes) NOAA owns is real-time collection, quality control, product synthesis, and analysis of a remote sensing pipeline (process) for creating and serving authoritatively certified precipitation data. This pipeline

(process) relies on one and five minute report output collected by individual sensors that are multi-agency (multi-DTE-user) owned and operated as part of a national in-situ sensor network. Sensor data are collected and combined with other data to perform internal quality control, and the products that are produced are ultimately published for use by other DTE users through several types of interfaces in various formats to make decisions with broad international legal and regulatory ramifications. Similar to DMME, NOAA relies on multiple algorithms of various nature to maintain the precipitation pipeline, including multi-tier quality control algorithms that use historical data and machine learning based anomaly detection. Like DMME, the QC algorithms, and those important to other parts of the pipeline, depend on access to supplementary data for analysis and improvement, ultimately resulting in support of downstream decisions that have broad implications for the health of the earth system itself.

Both of the DTE users described in this example do related yet obviously different work in very different contexts, and both have different viewpoints in terms of decision support, but it is fairly obvious that each may benefit to their own ends from automated access to each other's data for improving their own models, quality control, predictions, products, and reasoning.

To enable this data sharing, we provide a DTE-FS compliant DTE-F to both users, and they thus become full participants of the resultant user-driven DTE. Each user selects or constructs a DTE-F compliant API to interact with the system. When initializing as users of the DTE, they are loaded as known entities, choosing their defaults about access rights and methods of acquiring access tokens. Once users of the system, they each describe their individual process through the API as a set of tasks based on the traditional pipeline, where each task is described in terms of its workflow and output structure.

The output structures for each task are constructed according to each user's own unique understanding of their own data, but with each backed by the DTE-F implemented reference model, the output from each task is semantically interoperable and discoverable by both themselves and other users, both preserving lineage analysis and promoting process, method, and/or ontological reuse. Every output is intrinsically described in terms of schematic and semantic representation, preservation information, potentially customized access rights over the default values, and packaging, and each gets a searchable and findable description. Tasks are purpose specific and linked in order, with input tasks describing the process of retrieving or accepting data, identity tasks describing individual entities, transformation tasks describing what shape the transformation produced data in, and how to understand it.

When tasks are linked to the process, the process is similarly given a description on accessibility and packaging, and when complete, it is deployed. The deployment validates the process in terms of task linkages, and if valid, makes the process, process tasks, and archetypes used to define process task outputs available for search as structure. This is accomplished by other automated DTE users that listen for new process deployments. One of these users looks for new archetypes, one of them looks for new processes, and both run NLP driven processes that store and maintain a hierarchical knowledge of methods for drilling down and accessing a particular federated DTE user's data.

Upon deployment, both user processes are active, able to handle event driven data from sources specified by individual processes, or alternatively they may begin to go out and retrieve data in an input task with custom configuration. As the processes work, they begin executing computational logic according to workflow, and persisting archetype-derived data entities, which are now available for interoperable search and discovery.

One or both users might then decide to augment their own process, and to that end, search the DTE-F API for archetypes with labeled fields they are interested in, processes that might have description they are interested in, or templates that might have context they are interested in. In the NOAA user's case, once it discovers the output template from a given data producing task owned and performed by DMME contains rainfall data for a specific area, in a specific format, it can understand how to acquire this data, and creates a new version of its own existing process that listens for triggers on completion of the DMME rainfall data task, assesses it in terms of its own online historical accuracy score model of that metric, and if it is satisfactory, integrates it into its own precipitation QC model, which it now generates and delivers multiple versions of, for selection and use by other external DTE users based on reference to the DMME identified rainfall system property.

Reference Implementation

The NOAA National Centers for Environmental Information (NCEI), part of the National Environmental Satellite, Data, and Information Service (NESDIS), is responsible for archival of all of NOAA funded data in a way that is consistent with National Archives and Records Administration (NARA) guidelines, and NOAA access to research results (PARR) guidelines. This means that NCEI must provide timely archive and broad access to data that comes from, among other sources, nationally scoped in situ networks, polar and geosynchronous satellites, focused research studies, ocean dives, autonomous vehicles, hurricane trackers, and more, and deliver this data in many specified ways to internal users, national and international partners, and the general public.

Efforts to fulfill this mission have taken on a large number of forms as the mission boundary conditions - the technologies, standards, agencies, people, and systems - have evolved over time. With the maturation of the cloud computing era, the method of mission fulfillment of NCEI has changed direction yet again, and the agency is currently building NOAA's Next Generation Cloud Archive (NAAS) service to meet the new implementation requirements and opportunities afforded by this development.

Eventually, NAAS is intended to function as part of the NESDIS Common Cloud Framework (NCCF), a holistic model based cloud computing system that is intended to function as a single unit toward fulfilling NESDIS goals. To that end, the NCCF includes aspects of data onboarding, product generation, ingest, and the NAAS. As the Archive and Access component, NAAS is in a unique position relative to the NCCF in two ways.

First, it must handle all NOAA funded data. This includes, but is not limited to, satellite data, which is historically the primary focus of NESDIS by a large amount. This necessarily means that NAAS sees and must support, in terms of content and character, an enormous amount and variety of data. Second, as the outgoing interface, NAAS must serve this data, to all intended users, in all expected ways. In the modern age, this involves knowing about and supporting all of the historically expected and held heritage formats, proprietary data repositories, custom schemas, and narrowly understood and used products that NOAA has accrued over the life of its mission and even the life of historical earth data record keeping, while simultaneously supporting things like distributed format transformations, on-the-fly aggregations, and targeted catalog platforms, including NASA's Common Metadata Repository (CMR), to keep up pace with and stay compatible with and useful to modern machines and users.

Based on these boundary conditions, it is clear that goals of the NAAS readily fit into the DTE model as described above in the DTE-FS. While 'only' the Access and Archive part of the NCCF, as both a consumer and disseminator of all NOAA product data, the NAAS is clearly an ideal user of a DTE, both for its own benefit and that of its intended consumers. To this end, the design of the NCCF Archive and Access service essentially 'trained' the DTE-FS model against specific NOAA user community requirements and boundary conditions, including reference model ideals, NCCF cloud architecture guidance related to implementation on Amazon Web Services, and governance structure.

The DTE-F developed for NCEI's use in building the NAAS is designated at the virtual Archival Information Package (vAIP). The name itself belies one of the core choices of DTE-FS implementation, which is the reference model used in defining the ontology to be used for defining all system concepts through archetype, thus enabling intrinsic semantic interoperability through a small, static, space-complete, searchable model. This reference model, the Open Archival Information System (OAIS), is an ISO standard and stalwart pillar of archival thought, and a space systems recommendation. While extraordinarily large if taken on the whole, the only parts of the OAIS that were found to be needed for implementation to support a full DTE-FS implementation were a rather small collection. In fact, the NAAS took only the OAIS concepts of the information object, the information package, and the access aid, to construct its core ontology.

This ontology was then built and validated in OWL and RDF using visual tools including Protege, WebProtege, and VocBench. Figures 1 and 2 provide a WebProtege visualization of the primary concepts of concern on which all other concepts may be implemented as archetypes.

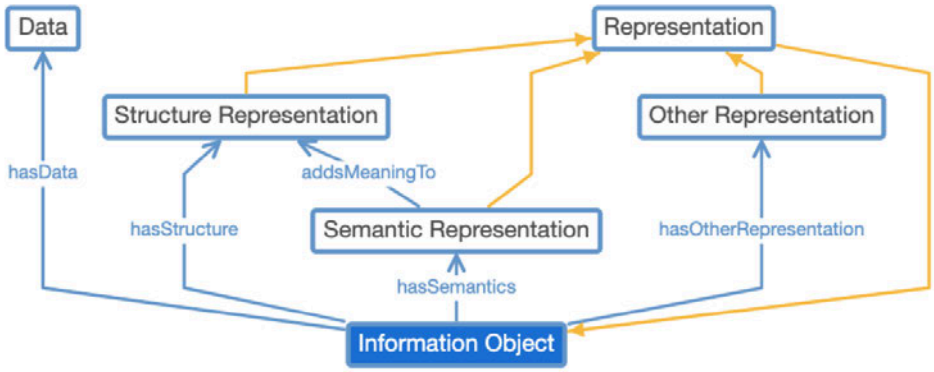

Figure 1*: the primary vAIP building block, the Information Object. Important to understanding this concept is that the data in an information object is the primary focus, held as 'just bits', and must be entirely qualified through its representation network, which is similarly made of information objects. This allows construction of rich semantic networks that are made easily machine-readable through SPARQL and easily machine-validatable through SHACL.*

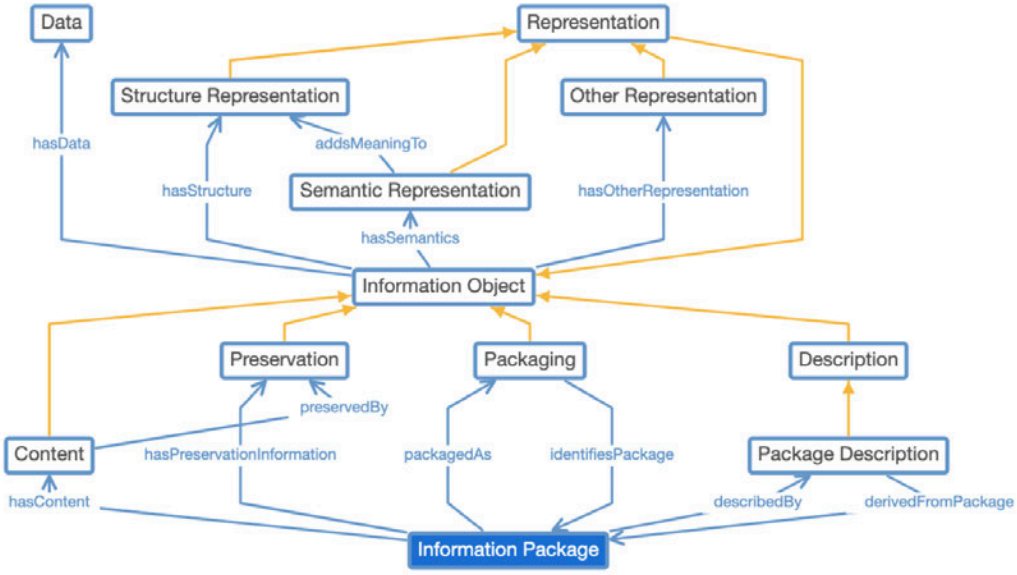

Figure 2 *- the secondary vAIP building block, the Information Package. Note that this ontological layer constructs a specific pattern of information objects, requiring storage of any unit to include its content, preservation (of which there are multiple types), packaging, and description information.*

Using these core concepts, following DTE-FS requirements, other fundamental aspects of vAIP DTE-F were implemented as archetypes. Figure 3 illustrates what vAIP implemented to fulfill the DTE-FS specification for a process task archetype (called step function archetype in the vAIP).

This archetype defines tasks that may be linked together in a DTE-FS Process, and it holds reference to its workflow configuration, output structure template, and the process flow policy (PFP).

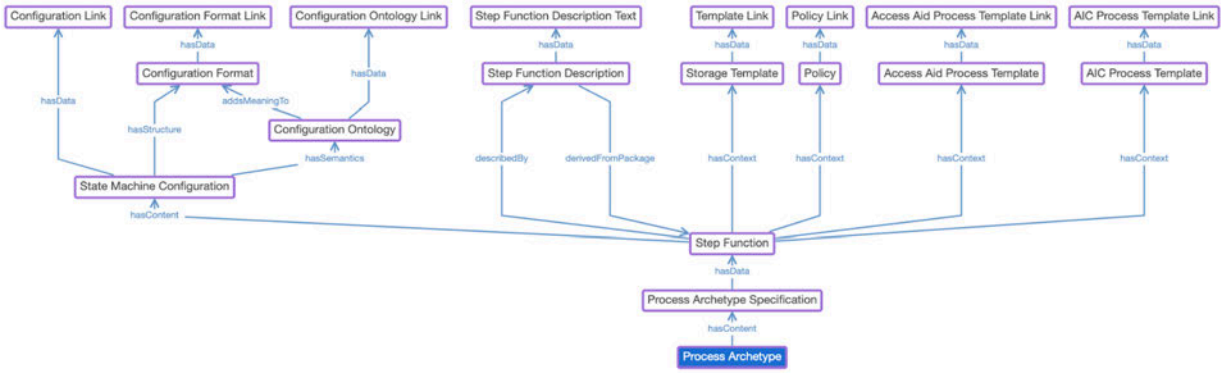

Figure 3: *A DTE-F task archetype, implemented in vAIP as a step function archetype, following the implementation choices of AWS step functions for workflow management.*

Each of the task types was also implemented in terms of structure in the vAIP ontology as a purpose-formed pattern. Figure 4 provides an example of an identity task storage archetype of a ‘granule’ implemented in terms of the OAIS concept of an identity focused archival information unit (only a cropped part of the archetype is displayed for compactness); Figure 5 provides an example of a transformation archetype, implemented using the OAIS concept of a membership-focused Archival Information Collection; and Figure 6 provides an example of an output archetype, implemented in OAIS as a Dissemination information Package (DIP) to an Access Aid (AA).

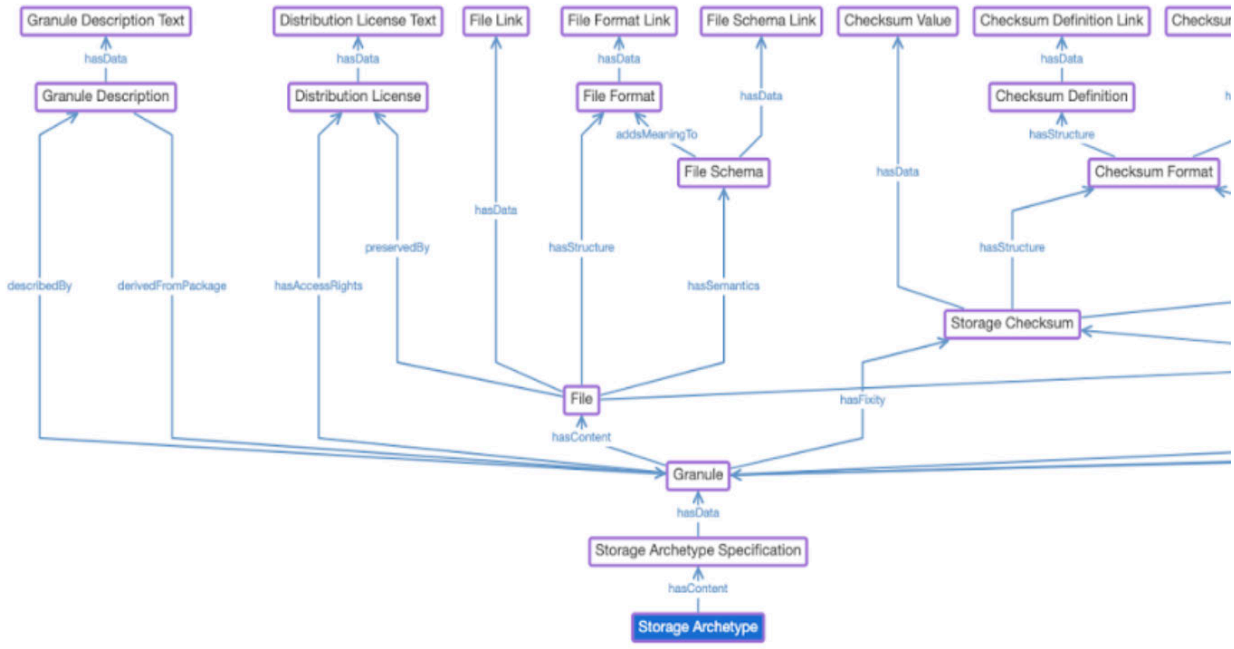

Figure 4: *An example of a DTE-FS identity archetype, modeled in the vAIP as an identity focused AIU.*

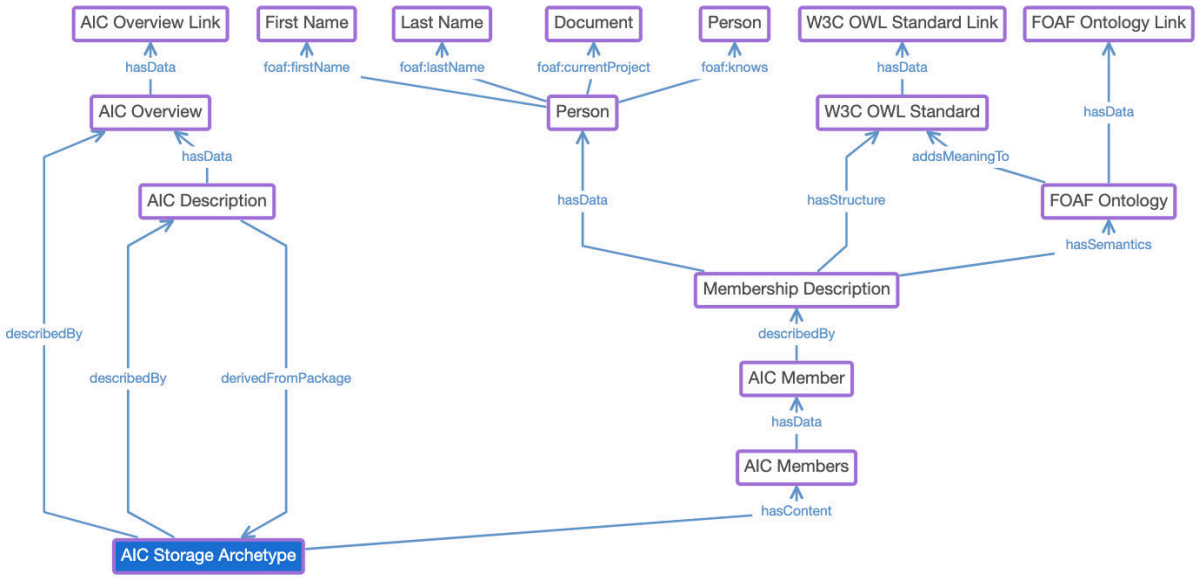

Figure 5: A DTE-F transformation archetype structure, implemented in the vAIP as a membership-focused AIC.

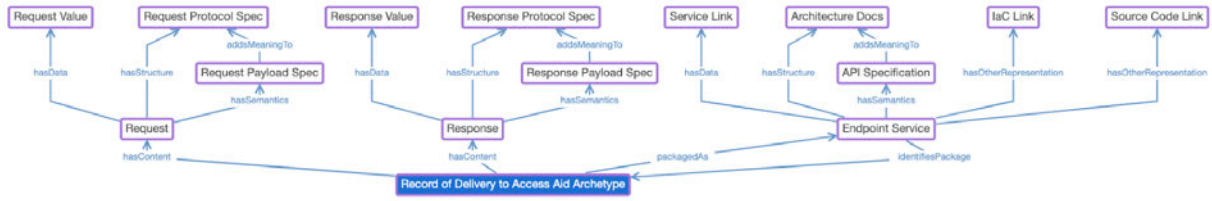

Figure 6*: A DTE-F output archetype structure, implemented in the vAIP as a delivery focused DIP.*

Importantly, these archetypes are only examples. The vAIP API both satisfies and allows other DTE-FS requirements to be satisfied in terms of user-specific, human and machine readable, findable and accessible, semantically interoperable, configurable, reusable, and versionable archetypes.

In the vAIP, syntactic interoperability requirements and goals including those related to submission of entire requests, retries, provenance, and model-based syntax, were met by implementation of several ‘workflow’ archetypes in the AWS step functions WMS, and functional requirements surrounding real-time data and submission were handled through their connection to the AWS eventbridge event bus utility. The workflow archetypes themselves were stereotyped according to the task types, and each includes automatic wrapping utility around custom user-task code to handle automated schema matching. Figure 7 represents the input, identity, transformation, and output workflow archetypes. As they are built in AWS Step Functions, their modeled design exactly matches the structure of implementation itself. Note the hidden model layer of the DTE-FS implementation used to trigger the overall process from real-time messages about data events.

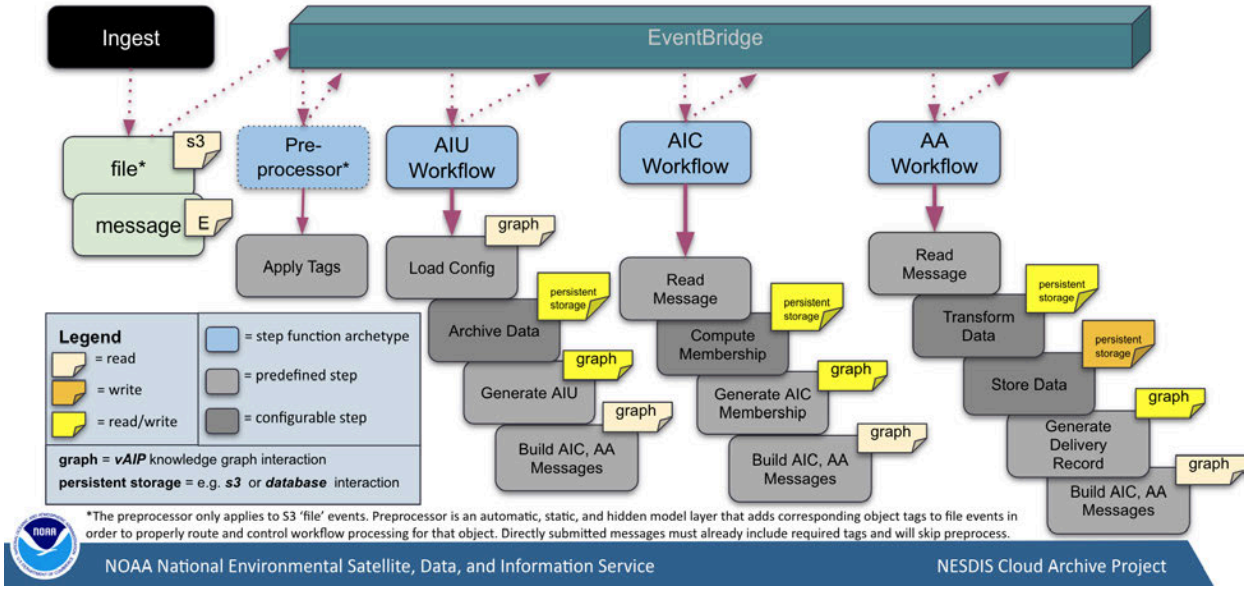

Figure 7: *Workflow archetypes implemented in the vAIP to satisfy, from left to right (ignoring the preprocessor), DTE-F task types of identity, transformation, and output, respectively.*

Further note that functional requirements of a DTE related to syntactic interoperability and non-functional requirements related to concurrency and composability are satisfied through the use of the EventBridge event bus as an intermediary between each workflow. As stated previously, validation of tasks is handled automatically based on key-value pair matching, and may eventually be improved further through the use of Natural Language Processing (NLP) reasoning to support semantic evaluation of workflow output to persistence input, and persistence output to downstream task input.

Task composition in vAIP is handled through implementation of the DTE-FS concept of process as a Data Stream Configuration, or Process Flow Policy. These policies further meet requirements of archetype description and versioning through their use of the process flow policy to store themselves as units within the system itself. Figure 8 illustrates composition of the policy, while figure 9 illustrates deployment of the policy itself within the vAIP framework.

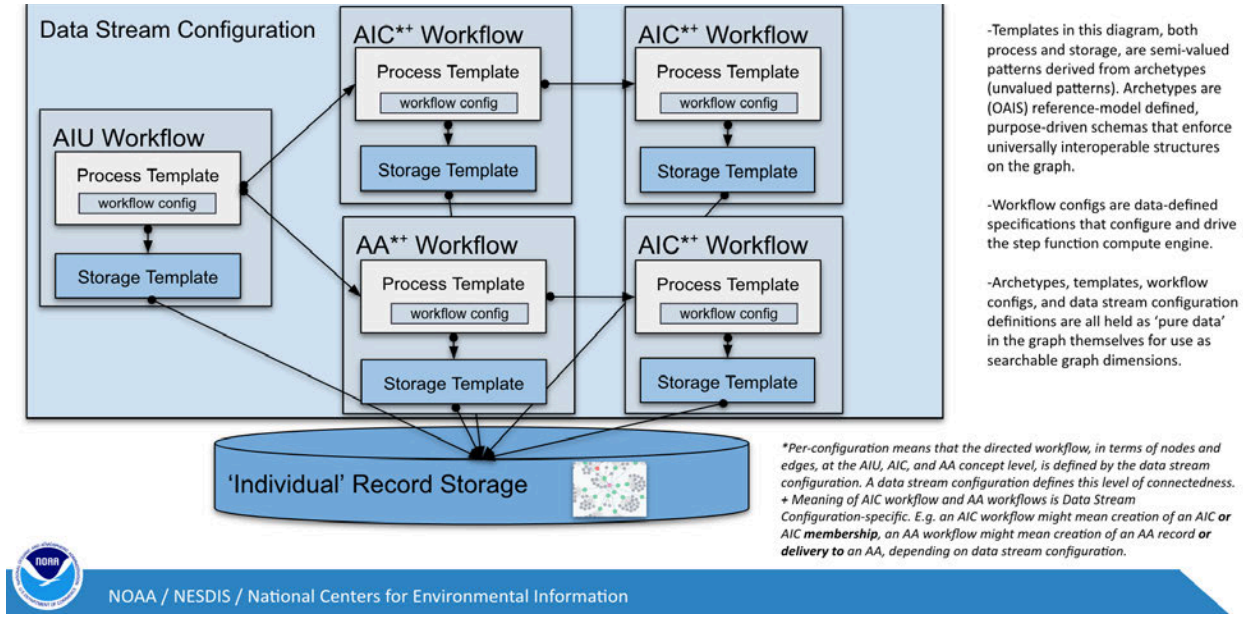

Figure 8: *The DTE-FS concept of process, along with notion of versioning and access requirements, as implemented in the vAIP as a 'Data Stream Configuration'.*

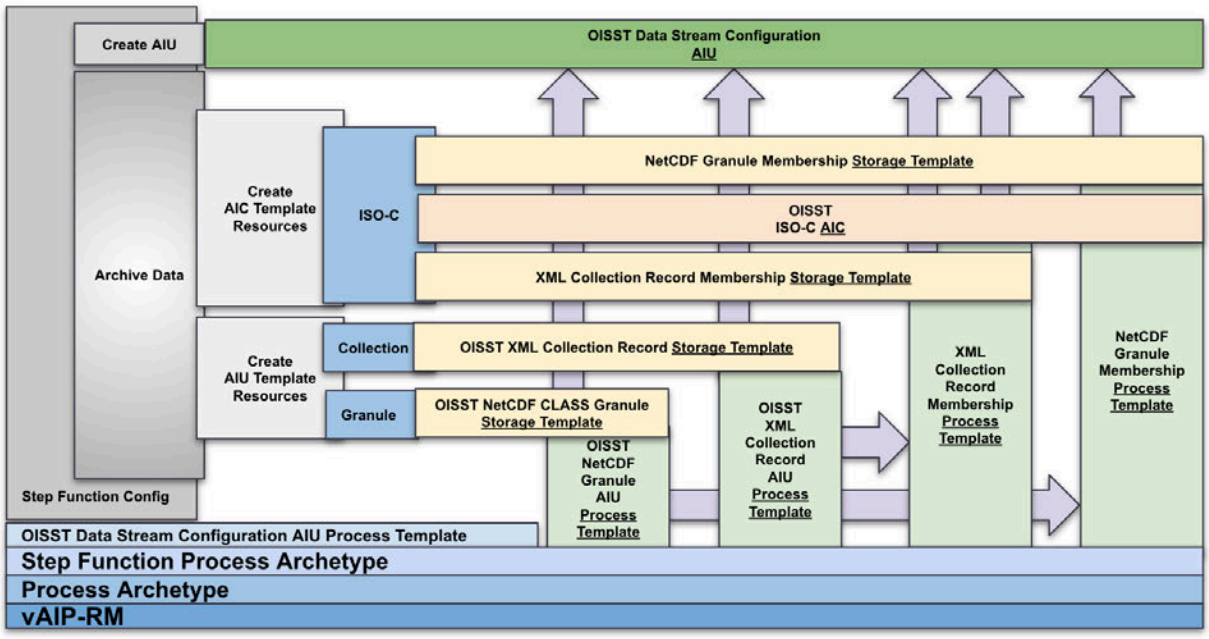

Figure 9: *Satisfaction of the DTE-FS requirements related to machine and human accessibility and versioning of processes within the archetype system as implemented in the vAIP.*

DTE-FS schematic interoperability requirements were demonstrated as partially satisfied through vAIP implementation of process triggering and task execution, while schematic interoperability between task output structures is further illustrated in figure 10, which also illustrates how identified entities within the system, defined in terms of rich contextual networks of their own, may be made semantically available in transformation contexts.

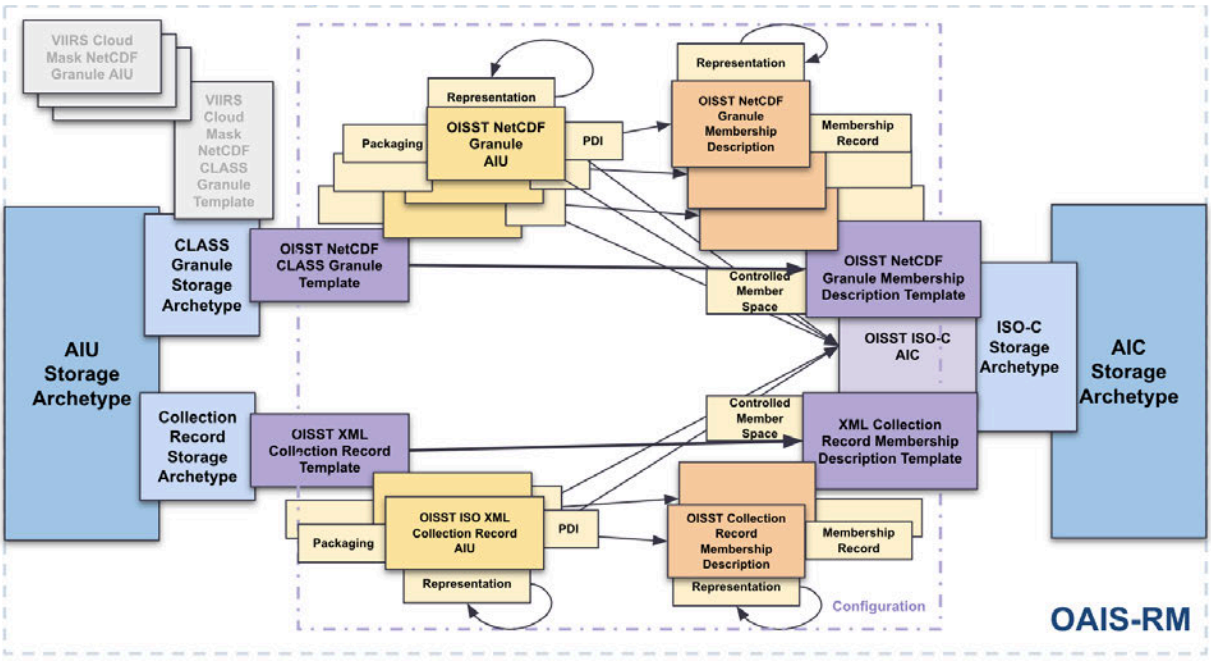

Figure 10: *Visual illustration of task to task structure in terms of schematic and semantic interoperability.*

As a DTE-F based on DTE-FS, the vAIP is intended to satisfy a broader DTE. The workflow for how this should look is illustrated in figure 11. Figures 12 and 13 illustrate how vAIP satisfies DTE-FS requirements of the API, in terms of provision and use as a building block.

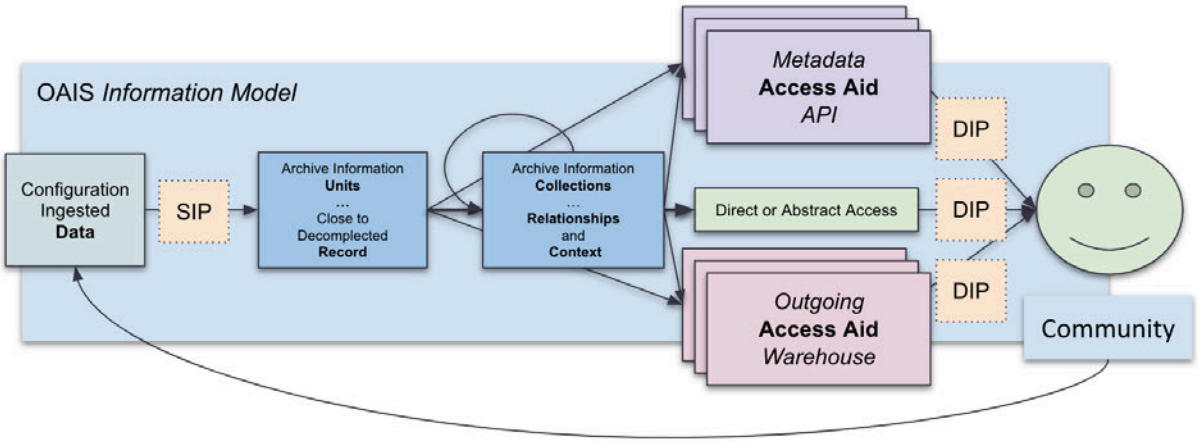

Figure 11: *Concept of Operations of the vAIP as an enabler of NOAA as a DTE user.*

```python
import vaip

#Create a new NEXRAD configuration
my_nexrad_configuration = vaip.create_configuration()

#Create AIU workflows for each data type in the configuration
tarball_aiu_workflow = my_nexrad_configuration.create_aiu_workflow()
xml_file_aiu_workflow = my_nexrad_configuration.create_aiu_workflow()

#Note that in the future, we can support manual workflows with 'human-in-the-middle' activities.
my_human_workflow = my_nexrad_configuration.create_manual_aiu_workflow()

#See if we have any available AIU archetypes that might match our criteria

matching_aius = vaip.find_aius(
    filters={"name": ["NEXRAD", "Granule"], "representation": ["File Link"]})

# zero, there are no matching AIUs in our system with these filters
print(len(matching_aius))

#We now need to create AIU storage for each of our workflows
tarball_aiu = vaip.create_aiu()
xml_file_aiu = vaip.create_aiu()

#Let's get a validation report to see what's required for our model
# returns a SHACL validated map and overall status (FAIL)
validation_report = vaip.validate_aiu(tarball_aiu)

#Now let's fill in our tarball archetype with some values
packaging_info = tarball_aiu.add_packaging_info(
    name="object_prefix", type=vaip.LINK)
structure_rep = packaging_info.add_structure_representation(
    name="object_prefix_format", type=vaip.LINK)
semantic_rep = packaging_info.add_semantic_representation(
    name="object_prefix_layout", type=vaip.LINK, structure_representation=structure_rep)
```

Figure 12: *Screenshot of use of a space-complete API for encapsulation of DTE interoperability for the purposes of security, usability, and sustainability.*

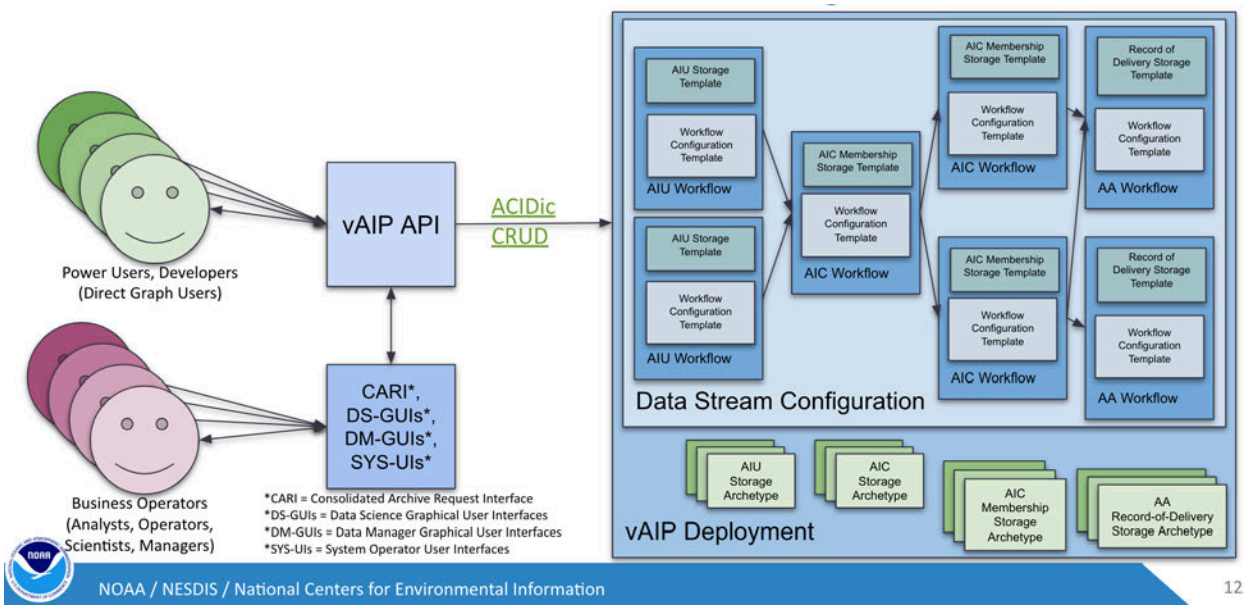

Figure 13: *Notional expansive use of the vAIP API to support broader user needs in terms of visualization tools and interfaces.*

References

Communicating sequential processes - Wikipedia
Onyx Platform Information Model
Actor model - Wikipedia
ORE Specification - Abstract Data Model
PROV-DM: The PROV Data Model
Reference Model for an Open Archival Information System (OAIS)
Reflections of knowledge | Ruben Verborgh
Semantic Sensor Network Ontology
A curated, ontology-based, large-scale knowledge graph of artificial intelligence tasks and benchmarks | Scientific Data
A Survey on Ontology Evaluation Methods
Why openEHR is Eating Healthcare. It is just over ten years since Marc… | by Alastair Allen | Medium
Noninvasive Re-architecture of Legacy Systems
Harness - A State Testing Environment for Big Data Algorithms
TaskAPI - A Declarative Process Data Model and API for Distributed Systems
GitHub - zeus-volkov-systems/messageapi: Declarative Polyglot Rules Engine
Mostly Abstract REST Structures
Simulation vs. Machine Learning - Vortarus Technologies LLC
What Is a Digital Twin? - MATLAB & Simulink
Actors and the Process Calculi: A Comparison | by Stephen Bly
Architectural Frameworks, Models, and Views | The MITRE Corporation
An Introduction to Model-Based Systems Engineering (MBSE)
SQALE - Wikipedia
The FAIR Data Principles - FORCE11
CARE Principles of Indigenous Data Governance
Ontology Summit 2020 Communiqué: Knowledge Graphs
5 Star Linked Data - Government Linked Data (GLD) Working Group Wiki
NESDIS Cloud Strategy
PB-23-11_Revised GEO Data Management Principles Implementation Guidelines.pdf
https://www.usajobs.gov/job/665456900
https://www.dlib.org/dlib/june06/chan/06chan.html
https://www.iso.org/obp/ui/#iso:std:iso-iec:21838:-1:ed-1:v1:en