Accelerating Data-Driven Discovery With Scientific Asset Management

Robert E. Schuler
schuler@isi.edu

Carl Kesselman
carl@isi.edu

Karl Czajkowski
karlcz@isi.edu

Information Sciences Institute
University of Southern California
Maria del Rey, CA 90292, USA

***Abstract*— Current approaches for scientific data management have failed to keep pace with the needs of increasingly data-intensive science. The overhead and burden of managing data in complex discovery processes, involving experimental protocols with numerous data-producing and computational steps, has become the gating factor that determines the pace of discovery. The lack of comprehensive systems to capture, manage, organize and retrieve data throughout the discovery life cycle leads to significant overheads on scientists time and effort, reduced productivity, lack of reproducibility, and an absence of data sharing.**

In "creative fields" like digital photography and music, digital asset management (DAM) systems for capturing, managing, curating and consuming digital assets like photos and audio recordings, have fundamentally transformed how these data are used. While asset management has not taken hold in eScience applications, we believe that transformation similar to that observed in the creative space could be achieved in scientific domains if appropriate ecosystems of asset management tools existed, tools to capture, manage, and curate data throughout the scientific discovery process. We introduce a framework and infrastructure for asset management in eScience and present initial results from its usage in active research use cases.

I. Introduction

Scientific discovery is undergoing a profound transformation driven by high-throughput instruments, pervasive sensor networks, large-scale computational analyses, growing dependence on collaborations and virtual organizations, and a changing scientific communications ecosystem that places increased emphasis on data sharing. Discovery driven by computational analysis and exploration of data has been recognized as the fourth paradigms of scientific discovery alongside empirical and theoretical methods of discovery, leading us to a new era of *Discovery Science*, a methodology of developing and testing hypotheses from large, rich, and complex data where the initial observation may precede the hypothesis [1].

Traditionally, a knowledge cycle in scientific discovery has been regarded as the formation of new knowledge through repeated turns of hypothesis, prediction, observation, and analysis, which are punctuated by transmission of results in the form of publications [2]. In the new paradigm of discovery science, these knowledge turns are increasingly dependent on a scientists ability to acquire, curate, integrate, analyze, and share data well beyond the compact reports offered by traditional publication. While the details of these cycles vary from domain to domain, and indeed across time within a single discovery process, data driven discovery cycles share common characteristics and face similar data related problems regardless of the domain. Experimental protocols involve multiple steps with data capture from diverse, high throughput, high fidelity instruments, analytic pipelines or computational simulation models. The results of one cycle of experiments are iteratively fed back into the system to refine the next series of experiments. Data must be contextualized within the protocol steps they are captured and may be related to physical specimens and other details of the experimental stage that produced the data. Data will also be produced by computational and human analyses in other steps of the protocol and also must be contextualized and linked to related data.

As the complexity and volume of the data that are used to drive knowledge turns in eScience applications increase, the ability of the scientist to manage the logistics of executing these cycles can become a rate limiting step. Current approaches to scientific data management have failed to keep pace with the needs of increasingly data-intensive science. Managing data is often done manually with "meaningful" file names and directory hierarchies, locally coded spreadsheets, and ad hoc laboratory notebooks. As noted by [3], "large amounts of data are generated using a variety of innovative technologies and the limiting step is accessing, searching and integrating this data." Scientists and other data analysts report spending 50% or more of their time on data wrangling (locating, extracting, cleaning, formatting, organizing) tasks [4], which leads to potential misinterpretation of results, misuse of data, forgetting pertinent details to describe the data, inability to find and retrieve previously captured data and results, and other similar issues. Concerns over repeatability of scientific results are growing along with an increase in scientific retractions [5], and some reports indicate that there is as little as 10% reproducibility of scientific results which cite lack of data publication as one of the significant factors [6]. As noted in [7], tools merely to support data capture are "just dreadful" and "we lack good tools for both data curation and data analysis." These sentiments echo similar observations made decades earlier by J. C. R. Lick-

The work presented in this paper was funded by the National Institutes of Health under awards 5U54EB020406, 1R01MH107238-01, 5U01DE024449 and 1U01DK107350

lider in his seminal work on the man-computer symbiosis, as he states, "...my choices of what to attempt and what not to attempt were determined to an embarrassingly great extent by considerations of clerical feasibility, not intellectual capability [8]." Decades later, these same issues continue to obstruct data-driven discovery.

While scientific data management has lagged behind the needs of data-driven discovery, this is not the case in other creative activities that also depend on managing large, complex data sets. For many years, the prevailing method of collecting digital pictures was to categorize images following user-defined, ad hoc naming conventions, resulting in a rigid, hierarchical, and often-confusing array of files and directories, much like scientific data is managed currently. Today, photographers use em digital asset management (DAM) systems such as Apple Photos or Google Photos to automatically discover and catalog digital images on their cameras or hard disk drives; extract metadata from the imported media; add user annotations; organize pictures into virtual collections (i.e., photo albums); browse and search on metadata, annotations or features such as faces in the picture; export data for manipulation by external photo editing tools; and publish to cloud-based sharing or printing services. Scientific data management by comparison has failed to address the data wrangling tasks incumbent on the majority of scientists today.

The ecosystem provided by DAMS streamlines the acquisition, management and sharing of digital assets such as pictures and music, and has had a transformative effect on how we think about and use these data. We assert that creating a digital asset management ecosystem for eScience data could have a similar impact: to transform the way that scientists interact with data, facilitate data-driven discovery and radically hasten knowledge turns for scientific discovery. An eScience DAMS ecosystem should provide the following capabilities:

- Acquisition and characterization of diverse scientific assets, including experimental data from instruments (e.g. microscopes, sequencers, flow cytometers), outputs from computational models, and results from analysis pipelines. Data must flow freely and automatically from the points of production into the management system, much like pictures flow from a smartphone into a management application.
- Model-driven organization and discovery of assets. Successful DAMS systems in the consumer space provide end users with intuitive and interactive ways of organizing and discovering assets that may be related via a complex underlying model. For example, in music DAMS systems, one may discover based on artist, group, work, composer, instrumentation, genre, etc. Similarly, an eScience DAMS should provide model based organization and discovery, in spite of the fact that the models may vary radically from domain to domain, may cross domains (multi-disciplinary collaborations) and vary over time as the discovery process unfolds.
- Storage and retrieval of eScience data assets. These assets may be very large, and may be physically distributed in local, enterprise, and cloud based storage systems.
- Aggregation and exchange of data collections. An eScience DAMS should be viewed as the hub of a data management ecosystem and not create unnecessary data silos. Hence, it is critical that the DAMS allow users to assemble and export data sets for consumption by other tools and users.
- Rights management/access control. A core function of DAMS is management of IP associated with assets. In the eScience environment, this means enforcing data use agreements, access to proprietary data, time driven data embargoes, and different user roles within and across collaborations.

This paper makes the following contributions:

1) We introduce the idea of asset management as a way of addressing eScience data management challenges.
2) We present the requirements for asset management systems based on analysis of eScience applications.
3) We propose a general architecture for scientific asset management ecosystems.
4) Finally, we describe the first platform implementation for scientific asset management.

In the following sections, we will address each DAMS capability. We will then describe our proposed architecture for scientific asset management and the implementation of DERIVA, a platform for introducing DAMS functions into eScience applications. We then share a preliminary evaluation of the approach and its framework in the context of science applications that use it regularly. Finally, we present plans for future work and then offer our conclusions.

II. CHARACTERISTICS AND REQUIREMENTS FOR SCIENTIFIC ASSET MANAGEMENT

Scientific discovery must be supported by an ecosystem of services and tools – instruments, computational workflows and analysis pipelines, domain repositories and data hubs. In scientific discovery, as much or more than any other discipline, the data are what is important and will outlast the systems used to interact with the data. Here we describe the characteristics and requirements for scientific asset management.

A. Acquisition and Characterization of Scientific Assets

We represent data driven discovery in terms of an evolving set of *scientific "digital assets"* (i.e., research data or simply assets), which are described and related to one another via a *domain model*. The idea of an asset contextualized within a domain model is similar to semantic models for describing argument and evidence as "micropublications" [9] where the continuous acquisition and characterization of assets throughout the discovery process may be viewed as incremental micropublications with each asset as an embodiment of evidence on which scientific arguments may be grounded.

Diverse sources of assets. An asset may be generated by sensors, instruments, or as the result of a computation.

Like photos on a smartphone, assets should be seamlessly integrated into the asset management system. The production of assets is itself fluid and cyclic throughout the discovery process as new data are generated, analyzed, shared with collaborators, and exported for publication. As new assets are acquired, they may be immediately consumed by other actors such as collaborators or computational workflows and analysis pipelines.

Diverse forms of assets. Scientific assets come in different forms, formats, and conceptually at different levels of granularity. For example, a video recording can also be represented more granularity as a time series of individual frames. In some cases, therefore, it may be useful to reference the parts versus the whole and conceptually to model the video as an aggregation of frames.

Incremental refinement of assets. The framework must allow incremental refinement of assets throughout the scientific discovery process. Data may be captured early in the discovery process. However, at the point of acquisition researchers may only have minimal contextual information to describe the asset, such as the date and time of capture, the instrument type and settings, and identity of the investigator. It is not until later that more information regarding the asset will be known, such as measures of data quality. The acquired data may also become the input for downstream computational analyses, which will generate new insights and may result in derived data of its own. Data does not enter the discovery process fully formed but often goes from a minimally- to maximally-described state [9].

Automation and self-service curation. Manual data entry has been noted as one of the key barriers to successful adoption of data management services [10]. The framework must enable self-service curation of assets with simple user interfaces and automated services to reduce manual effort where possible. However, the diversity of scientific domains and data and their unique requirements means that simple one-size fits all asset management applications will not suffice. At the same time, developing new interfaces and applications for each use case is prohibitively expensive and time consuming. The user applications must adapt to the underlying domain model. They cannot assume a rigid structure but must be flexible both to the structure of the data and also the workflow of researchers using the system.

B. Models and Evolution for Scientific Asset Management

Domain models represent the key concepts, behaviors and relationships of the participants and elements in a real-world system. Domain models are used to describe scientific assets in order to link, organize and contextualize them within the discovery process. Effectively modeling the information for such diverse and complex domains of science motivates the need for domain modeling approaches that support well-defined, structured information. Yet, the process of scientific discovery is always unfolding and yielding new insights and understanding of the domain and hence systems to support it must be able to evolve as well.

Structured data models. We argue that when one considers the complexity of scientific information and the importance of precise descriptions of research and results that a *structured* data model is necessary, ideally one based on a strong formalism such as relational or graph theory. For example, the *entity-relationship model* (ERM) can serve as the underlying meta-model for domain models in scientific asset management. ERMs are extremely expressive and can be used to describe data in tabular as well as graph structures through commonly used idioms. Another important factor in considering ERMs as an underlying meta-model is that the majority of scientific data is in practice represented in tabular structure [11] and therefore is a natural fit for use in scientific data modeling.

While it has become increasingly popular to abandon structured data models like ERMs in favor of so-called *NoSQL*, *schema-less*, *key-value*, or their decades old predecessor *entity-attribute value* (EAV) databases in order to avoid data modeling, there is nothing inherently static or restrictive about ERMs; in fact, most modern database management systems support schema manipulation. Scientific query workloads have been shown to depend on complex query operations not possible with simpler query dialects and less structured models [12]. More recently, traditional database systems have added support for features that were thought of as the domain of "document-oriented" databases, such as special handling of JSON and text processing, thus offering the features of relaxed schema with the advantage of having full capabilities for structured data management.

Evolution and introspection of models. Upfront data modeling is a significant bottleneck to the adoption and usage of structured data models. For science applications, it may be impossible to define a data model upfront and the model may change in unexpected ways throughout the discovery process. It is therefore essential that the ecosystem for managing scientific assets be based around generic tooling that *introspects* domain models and adapts to them. The systems must allow scientists to begin with minimally-specified models and evolve the model and data throughout the discovery life cycle, while continually refining and increasing the richness and rigor of the structures used to describe their research. Likewise, the ecosystem of tools and services for asset management must be sensitive to the changes in the model. Since data producers and consumers operate independently and asynchronously, they must adapt dynamically. Even in the middle of an interaction with the data, the model may change based on the interactions of another agent in the system. Tools should be model-agnostic so that they may be useful across diverse scientific applications and to allow model evolution without having to upgrade software repeatedly for every model or data change.

One key obstacle for supporting flexible yet structured data models, is that many applications developed in the popular object-oriented programming (OOP) paradigm are often developed using *object-relational mapping* (ORM) libraries that define their own proprietary query dialects and suffer from the well-known *object-relational impedance*

mismatch problem i.e., objects and classes map poorly to relations. Unlike ORMs, we take an approach that does not hide or obfuscate the underlying model from the client. Instead, *the elements of a domain model are represented and exposed directly and transparently through protocols and interfaces* appropriate to the technology platform in which asset management is implemented. This argument is similar to the "don't repeat yourself" or "DRY" principle – that information should have a single unambiguous representation. By contrast, ORMs and popular Web frameworks introduce additional, obfuscated layers that must be updated whenever the data model changes.

Extending and enriching models with annotations. While a formal meta-model, like the ERM, is necessary to model scientific domains, it is insufficient to describe the complete semantics of the model. In a relational model, one cannot describe anything more about a particular element in the model other than that it is a 'table' or a 'column' or a 'reference.' Additional semantics on the model are needed as a way of extending and enriching the basic concepts of the meta-model. For this purpose, *model annotations* are a way of filling the gap beyond what the meta-model can provide. The annotations are applied to various levels of the domain model including the individual schemas, tables, columns, and foreign key references. The annotations can be used to indicate additional semantics about an element of the model, provide presentation hints to the user interface for how to render a model element or its data in an interface, and other uses applicable to the domain or applications and agents involved in mediating and manipulating the schema. Examples of annotations are shown in Table I.

TABLE I
EXAMPLES OF ANNOTATIONS USED TO DESCRIBE MODELS.

Annotation	Description
tag:misd.isi.edu,2015:default	Schema or table to be selected by default by user interface presentation.
tag:misd.isi.edu,2015:url	Table or column should be rendered as an link.
tag:misd.isi.edu,2015:vocabulary	Table should be interpreted as a controlled vocabulary.

Heuristics for understanding models. While domain models will be extremely diverse especially from one scientific domain to the next, we have found that a small number of *heuristics* can be very powerful in enabling presentation and interaction without statically coding behavior into the applications and tools that operate over structured data models. Our heuristic approach makes some simple assumptions about the semantics of the model, particularly as it concerns rendering, navigating, and updating data in a complex domain model. For example, when rendering a detailed view of an entity (i.e., a row) from a table, we can apply heuristics that denormalize the rendering of the entity in order to contextualize with its "neighbors" in the linked data graph of relationships it forms with references to and from other entities in the ERM. This heuristic strategy can be applied generally across different models, domains, and with alternative meta-models as an approach to developing general tooling for working with asset management systems. Examples of heuristics are shown in Table II.

TABLE II
EXAMPLES OF HEURISTICS USED TO INTERPRET MODELS.

Heuristic	Description
Ignore auto-generated	Disable input for system-generated keys.
Extended record	When displaying a record, extend the record with entities joined by many-to-one relationships.
Nested entity	When displaying a record, show preview of entities joined by one-to-many relationships.

C. Storing and Accessing Scientific Assets

Some of the closest approximates to asset management storage systems (a.k.a., data stores) come in the form of version control systems and object stores that operate on data as atomic units and permit changes only through explicit versioning semantics. In general, conventional storage systems do not necessarily suffice for the needs of scientific asset management. Here we discuss the requirements unique to storing and accessing scientific assets.

Stable naming of assets. Assets and their metadata are referenced by name, and data names can be shared between actors by external means or embedded within other data, which may exist in the same or different data stores. Therefore, references to data are not necessarily under the control of the data store holding the referenced data. Rather, references may be asynchronously communicated between distributed parties. Thus, within the context of asset management, a data store must deliver certain guarantees: a reference to an asset should never become ambiguous, but rather should always denote one particular asset whether or not the data is currently available for retrieval; a retrieval operation should be unambiguous, with a consumer being able to determine whether they have successfully retrieved the denoted asset or have encountered an access error. In short, the retrieval of named assets should be atomic, consistent and stable.

Immutability of assets. Scientific processes depend on reliable acquisition and sharing of data in order to support reproducible results, thus once an asset has been acquired it must not be mutated. Consumers such as manual reviewers of imaging data or computational workflows that take complex data as input must be assured that the assets they consume are exactly the assets that were produced and shared by the data producing actors. A close corollary to stable names for assets (discussed above) is that assets must therefore be immutable, such that once generated no edits or other in-place changes may be allowed. It is more acceptable, though perhaps not ideal, for an asset to be *retracted* from the system just as publications may be retracted from the scientific literature. In these exceptional cases, a marker sometimes called a "tombstone" should be left in the asset's

place so that references to the asset may be resolved at least and consumers can understand why the asset is no longer available. In data preservation terms, immutability is often called data or file *fixity* meaning the contents of a file are fixed and cannot be changed.

Versioning of assets. At times, a new asset will be generated based on incremental changes starting from an existing asset and in such cases, these assets enter the system as new *versions* of an earlier asset. Versioning, however, must be done explicitly as a new asset with an explicit version relationship to another existing asset and as stated earlier must not mutate an asset in place. To assist in this a data store of assets should issue a name for an asset only once and may issue *version-qualified names* to add convenience to the consumers and references of assets.

Linking and provenance of assets. Versioning and other operations that derive one asset from another further motivate the need for facilitating models of provenance. We see these as one natural application of the general requirement for linked data, where the relationship between data is semantically described in terms of the processes and actors that produced an asset and existing models of provenance [13] should be employed for this purpose.

D. Aggregation and Exchange of Assets

Methods must exist to extract collections of assets under management for consumption by external human or computational agents. Several alternatives exist for exchanging complex information via various forms of *data aggregation packages*, for example Research Objects [14] or structured file formats such as HDF. These methods can be used to facilitate exchange of asset sets from a DAMS.

E. Policies for Managing Assets

Numerous forms of policy must be considered when managing scientific assets.

"Protected" data. Research involving sensitive data collect during studies involving human subjects are governed by policies administered by Institutional Review Boards (IRBs) or other bodies that set Data Use Agreements (DUAs). These data must be handled in compliance with rules governing privacy and strict data sharing limits. Under these circumstances, even when research concludes the IRB or DUA policies may stipulate how storage systems must be purged of stored information.

Private use of data. In general, data are of great value to the researchers that produce them. Data sharing embargoes specify limitations to access of data while the investigators, those that generated the data, use them in analysis and until they have been published. The ecosystem of tools for asset management must be cognizant of the policies, whether governed by formal policies or defined by individual researchers, and the tools must limit use and sharing of data in compliance with the specified policies.

III. Scientific Asset Management Architecture

Based on the requirements and characteristics of asset management from our analysis in the previous section, we have defined the scientific asset management architecture (fig. 1). The canonical architecture of the Web provides a blueprint for a community of active participants (users and automated agents) to interact by using simple universal service interfaces to manipulate passive resources on the Web, and we extend these patterns according to the unique requirements of scientific asset management to enable an ecosystem of tools and services for discovery science. Our goal is not to define an exhaustive collection of providers for various capabilities but rather to identify general classes of services and tools that are necessary for implementing asset management solutions in various configurations and for diverse scientific applications. We now describe the primary components of our scientific DAM architecture.

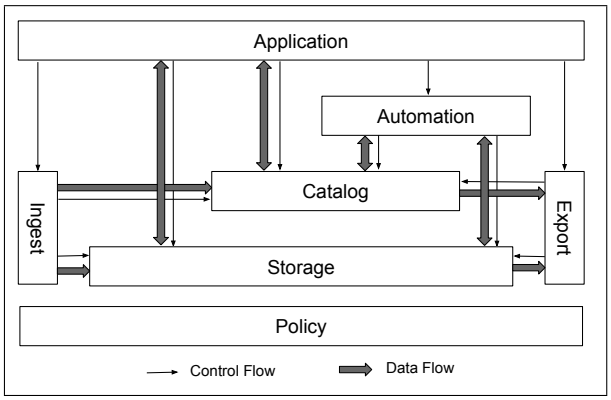

Fig. 1. Scientific Asset Management Architecture.

Application. Applications provide adaptive user interfaces for acquiring, curating, and consuming assets. They are model-agnostic and function by introspecting the data stores, in particular the structured data catalogs that hold the domain models and descriptive information. These applications support searching and browsing paradigms to help users locate assets of interest. They let users define custom subsets and slices of data and support entry and editing of new data and annotating of assets. Applications must be model-driven and reflect the current state of the catalogs and data stores without hard-coded assumptions and semantics about domain models.

Catalog. The Catalog layer represents specialized forms of the data store for recording, querying, and retrieving descriptive information (i.e., metadata) about scientific assets. As specified in the requirements, in order for the catalog to sufficiently handle domain models for scientific applications, it must support structured data models, model evolution, interfaces so that clients can introspect domain models, complex query operations and named queries, and model annotations. The catalog is primarily a passive service that applications and other tools (like ingest and export pipelines) interact with to store, query, and retrieve metadata about assets in the system.

Storage. The Storage layer represents the data store for bulk asset storage and retrieval. It must satisfy requirements for transactional update, permanent naming of assets, and non-destructive updates. It must ensure the immutability

or fixity constraints and may support versioning. Like the catalog, the data store is primarily a passive service that provides interfaces for applications and other utilities (like ingest and export pipelines) to store and retrieve scientific assets.

Ingest. The role of ingest utilities and services is to identify new assets of interest, extract and harvest metadata from them, record descriptive information so that assets may be contextualized with who, what, when and how an asset was generated, which may only be possible to do at acquisition. It must interact with catalog and storage services to record and store assets. The ingest layer may be configured offline or may be administered through applications.

Export. Export services or utilities, on the other hand, extract assets and metadata from catalog and storage services, serialize and package them in well-defined package formats, and may deposit them on recipient systems. As with the ingest layer, the services and utilities of the export layer may take an active (e.g., by polling) or passive (e.g., responding to events or other signals) role in identifying what and when to export assets. They may also be configured offline or administered online through applications.

Automation. A wide range of data management and manipulation services may be automated in an asset management system. The services and utilities of the automation layer represent a potentially broad array of capabilities that will be needed in scientific asset management. Automation services will primarily take an active role interacting with catalog and storage layer services to perform a variety of domain- and task-dependent operations, including but not limited to metadata extraction and harvesting, format conversion, image stacking, image tiling, down-sampling, applying compression to various content types, or indexing data.

Policy. The Policy layer represents a broad class of services, utilities, and other tools to help specify, evaluate, enforce, and otherwise make policy decisions. These tools tend to play a central and ubiquitous role in all manner of distributed systems. In asset management, the components of the policy layer may be used to express and evaluate policy rules for determining individual and role-based access to assets and metadata, for example. The proposed architecture does not specify or require that policy be implemented centrally or decentralized by the individual components of the different layers of the architecture which may make independent and local policy decisions in practice.

IV. DERIVA: A PLATFORM FOR SCIENTIFIC ASSET MANAGEMENT

Based on the above analysis, we have created a DAMS platform for eScience applications called the **D**iscovery **E**nvironment for **R**elational **I**nformation and **V**ersioned **A**ssets (DERIVA). The implementation of the DERIVA platform consists of a multi-tenant relational data service for domain models (ERMREST), a client library (ERMRESTJS), an object storage service for assets (HATRAC), a suite of adaptive user interface applications (CHAISE), a suite of utilities for ingest and export of assets and metadata (IObox), an asset aggregation package format (BDBag), and a shared authentication layer (WebAuthN).

A. CHAISE

CHAISE is the user interface application suite for DERIVA. CHAISE is implemented as JavaScript programs that dynamically generate adaptive, model-driven user interface applications for discovery, analysis, visualization, data entry, annotation, sharing and collaboration over scientific assets. CHAISE applications introspect and dynamically render relational data resources based on a small set of baseline assumptions, combined with its rendering heuristics, which may be influence, informed or overridden by model annotations defined on the domain models that are cataloged in ERMREST, and finally user preferences further refine the interfaces.

CHAISE is intended to support specific asset management interactions. As such, its presentation capabilities are narrowly scoped. CHAISE makes a few assumptions about how users will interact with the underlying data. A few representative but non-exhaustive examples of these assumptions include:

- Search, explore, and browse catalogs of assets;
- Linked data navigation between records;
- Add, edit, remove records from the catalog;
- Create, alter, or extend the domain model in the catalog;
- Subset and export collections of assets and metadata;
- Share collections with others;
- Annotate records with tags or controlled vocabulary terms.

CHAISE is structured as a set of programs that generate interfaces to search and browse assets using the powerful 'faceted search' paradigm (see fig. 3); deep drill-down, visualization, and 'linked data' navigation on individual records (see fig. 4); and record entry and edit. Internally, the CHAISE applications are built on AngularJS [1], a front-end model-view-controller (MVC) framework developed by Google, and a common library of routines used across CHAISE applications (see fig. 2). CHAISE uses the ERMRESTJS library, which provides JavaScript language bindings for the ERMREST protocol and a comprehensive set of APIs for working with the ERMREST relational data service.

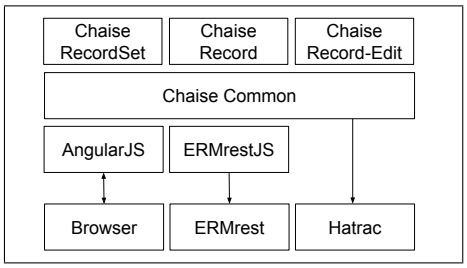

Fig. 2. Chaise layered architecture.

[1] https://angularjs.org

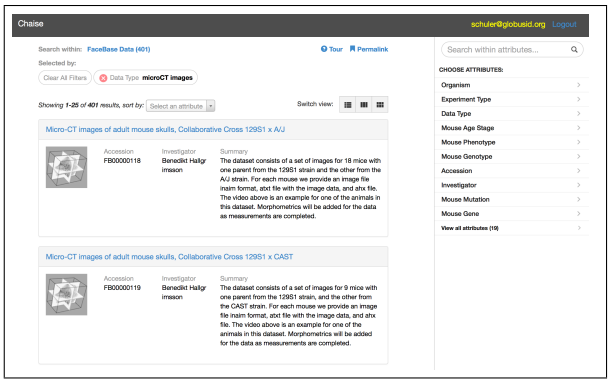

Fig. 3. Faceted search application.

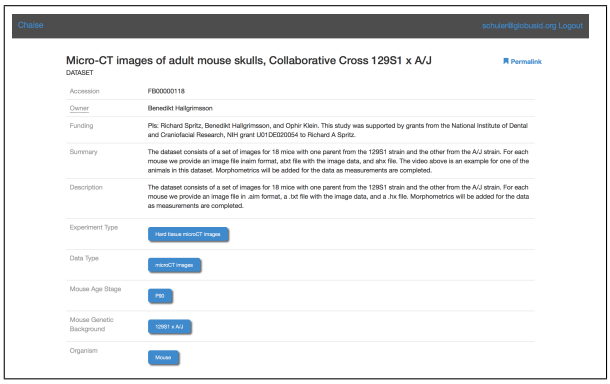

Fig. 4. Record details application.

CHAISE makes no assumptions about the structure of the underlying data model, such as its tables, columns, keys, and foreign key references. It begins by introspecting the data model by getting the schema resource from ERMREST. It then uses rendering heuristics to decide, for instance, how to flatten a hierarchical structure into a simplified (or "denormalized") presentation for searching and viewing. CHAISE then interprets the model annotations to modify or override its rendering heuristics, for instance, to hide a column of a table or to present a related entity as an embedded Web resource in an inline frame (`iframe`). CHAISE uses model annotations to determine when and how to integrate visualization tools into the display, including D3, Plot.ly, 3D volume rendering, and 2D, pyramidal, tiled image rendering.

B. ERMREST

ERMREST (Entity Relationship Model via Representational State Transfer) is a relational data service for the Web, and allows general entity-relationship modeling and manipulation of data resources by RESTful access methods. ERMREST serves as the metadata catalog of DERIVA and enables the evolving and dynamic data models needed for describing, contextualizing, and linking scientific assets.

The design goals for ERMREST were to enable non-expert users to create and evolve data models that represent the semantic concepts in their domain without the typical round trips from user to developer to database administrator and back. Many use cases can map into simple models with just a handful of entities and relationships and non-experts can easily think about their domain in terms of the main concepts that they want to manipulate [11]. By providing methods for incrementally creating these models, and by allowing users to express domain concepts directly in the catalog, it offers a platform where the data models can be created and maintained by the user community. While there will be situations in which a more formal up-front data modeling activity will be required and for which the full power of SQL may be needed, a significant number of important usage scenarios fall within the design space of ERMREST.

Our approach was to develop a service that supports resource manipulation idioms common to RESTful Web services. ERMREST maps *Entity*, *Attribute*, *Schema*, *Table*, *Column*, and other relational concepts to Web resources, which are referenced by Web resource identifiers (i.e., URIs). It supports the following interfaces:

- *catalog*: reference, retrieve, create, alter, delete "catalog" resources, each of which is an independent relational data store;
- *schema*: reference, introspect, and alter entity-relationship models (i.e., schema name spaces, table and column definitions, key and foreign key constraints, etc.);
- *entity*: reference, query, and manipulate entity records (i.e., rows of a table);
- *attribute*: reference, query, and manipulate projected attribute records (i.e., subsets of attributes of relations);
- *attributegroup*: reference and query projected attribute group records (i.e., attributes grouped by a subset of attributes of relations);
- *aggregate*: reference and query projected aggregates with supported *aggregate functions* such as count, min and max.

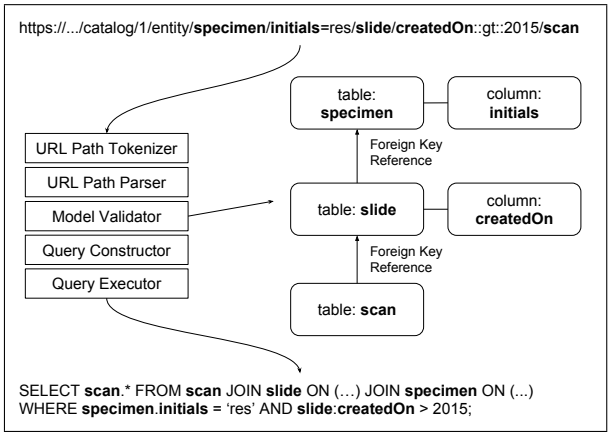

Fig. 5. ERMREST query processing example, showing the query URI (top), model definition (middle right), conceptual processing stages (middle left), and generated SQL statement (bottom).

The core of the implementation is in query processing. A request consists of the HTTP method (e.g., `GET`, `POST`, `PUT`, `DELETE`), URL path (e.g., `catalog/1/entity/scan`), and optional HTTP message body (e.g., JSON or CSV resource representation). The query language is a formally-defined context-free grammar in BNF notation and is parsed

using a generated "look ahead left-to-right" (LALR) parser. Fig. 5 depicts the query processing steps taken to satisfy each request to ERMREST.

First, the request handler tokenizes and parses the URL path into an Abstract Syntax Tree (AST) representation of the query. The catalog resource (e.g., `catalog/1/`) indicates which relational data store to address with the query. Next, the interface part of the path (e.g., `entity/` indicates which API will ultimately be used to answer the query. The remainder of the URL is what we call the *data path*, a pseudo-hierarchical expression that references, joins, and filters tables of entities and attributes. In the example, the first table in the path is the `specimen` table, and it is filtered by the binary predicate `initials=res`, which references the `specimen.initials` column in the left operand. The next URI component, which is delimited as usual by the '`/`' character, implicitly specifies a join with the `slide` table resource. The model validator will ensure that an unambiguous foreign key reference exists, in either direction, between the `specimen` and `slide` tables. Next, the filtered product of joined tables will again be filtered, this time by the binary predicate `createdOn::gt::2015` which filters slides that were created on or after year 2015. Finally, the last URI component again specifies an implicit join, in this case with the `scan` table.

The `entity` interface returns whole entities (i.e., rows of a table resource) and therefore does not require an explicit projection of attributes. With the `attribute` interface, however, URL *data paths* terminate with a list of attributes to be returned. More complex *data path* expressions are supported including projections involving multiple tables in the expression, joins between multiple tables at different depths of the expression hierarchy, table alias assignments and referencing, aggregation and grouping.

Finally, ERMREST recently incorporated row-level access control policies from its underlying PostgreSQL database engine. Combined with user- and role-based authentication from WebAuthN (discussed later), ERMREST can enforce fine-grain access control over the contents of the catalog. For example, permissions may be granted on tables based on group membership; visibility or editing may be enforced on a row-by-row basis, and more sophisticated policies are also possible.

C. HATRAC

HATRAC (pronounced "hat rack") is a Web service for storage and retrieval of scientific assets as Web resources in a RESTful service model. HATRAC treats scientific assets as generic, opaque, byte-sequences and may be viewed as an *object store* for asset management. It supports *atomic operation semantics* – that is, an asset is created and named, updated or deleted in an atomic operation where the operation either finishes and succeeds completely as expected or terminates the operation. Once an asset has been stored in HATRAC, it guarantees *data fixity*, first by enforcing immutability of stored objects, and second by maintaining check sum message digests which can be used to ensure data integrity. An "update" of an asset in HATRAC is non-destructive, such that the service preserves the current state of the asset while adding a new version of the asset in a *version-qualified naming scheme*. Similarly when "deleting" an asset, the service marks the named asset as deleted but does not release the name for reuse, so that it prevents names from being reused and potentially violating the *stable reference semantics* required by asset management. Finally, HATRAC supports a *hierarchical naming scheme* and allows users to define *access control policies* on assets by name and by subtree names in the name space to simplify management of access controls. At present, HATRAC supports two configurations: it may be deployed as a standalone server with assets stored on a local or remote file system; or it may be deployed on Amazon AWS with assets stored in the Amazon S3 object store.

D. IObox

IObox is a suite of modular utilities for ingest and export of assets to and from DERIVAand external data sources. Asset management for science must support diverse sources and formats. The utilities of IObox may be combined in a variety of configurations to support the unique requirements of different scientific applications. The utilities in this project are generally categorized in terms of extract, transform, or load (ETL) operations and uses the BDBag package format. In general, *extract* operations acquire assets from data sources (files or databases) and generate a BDBag package; *transform* operations alter the format or structure of the contents of a BDBag package; and *load* utilities take a BDBag package and upload the contents to catalogs and storage services.

At present, the utilities in the IObox suite include:

- *bag2dams*: takes a bag and imports it into DERIVA data stores;
- *dams2bag*: extracts metadata and assets from DERIVA data stores, serializes the contents, and generates a bag;
- *sql2bag*: connects to an ODBC-compatible database management server (e.g., Microsoft Access, Microsoft SQLServer, MySQL, etc.), executes user-specified queries, serializes the results and generates a bag;
- *xls2bag*: parses a Microsoft Excel spreadsheet and generates a bag;
- *xml2bag*: parses eXtensible Markup Language (XML) documents, serializes in tabular format, and generates a bag;
- *iobox-win32*: uses Win32 APIs to "watch" a Microsoft Windows file system, as files are generated asynchronously by connected instruments (e.g., microscopes, sequencers, etc.), it executes regular expression rules to identify files and extract metadata, and ingest directly to the DERIVA DAMS.

In addition, we have developed or integrated parsers for many important data formats for scientific assets, including but not limited to HDF5, NetCDF, CSV, Excel, TIFF, CZI, OME, NIfTI, DICOM, FASTA, FASTQ, and VCF.

E. BDBag

BDBag is a specification for asset aggregation packages. BDBag extends *The BagIt File Packaging Format (V0.97)* [2], incorporates the *BagIt Profiles Specification* [3], and adopts the *Research Objects* [14] semantic model for describing packages and provenance. The BDBag utilities are a collection of software programs for working with the enhanced specifications for bags. These utilities combine various other components such as the BagIt creation utility and the BagIt profile validator utility into a single, easy to use software package.

F. WebAuthN

WebAuthN is a compact, modular authentication provider framework written to support Python-based, RESTful Web services and is used by ERMREST and HATRAC. It allows deployment-time configuration of several alternative identity and attribute provider modules to establish client security contexts for Web requests by talking to a local or remote provider. The `client provider` is used when establishing the client (or user) identity while the `attribute provider` is used when establishing additional attributes, such as roles or groups associated with the client identity.

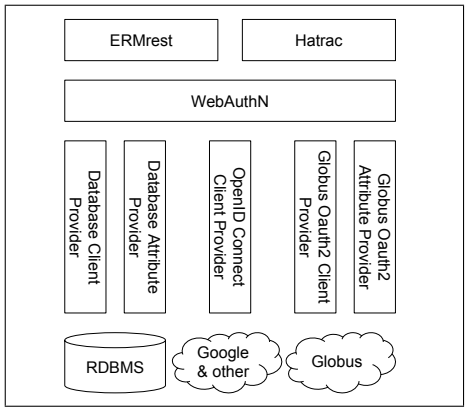

Fig. 6. WebAuthN layered architecture.

At present, WebAuthN is distributed with three identity provider (IdP) integrations:

- *OAuth2 OpenID Connect Provider*: any standard OpenID Connect IdP, such as Google, for client authentication;
- *Globus OAuth2 Provider*: Globus IdP which also supports delegated group management and access to numerous campus IdPs;
- *Standalone Database Provider*: standalone database IdP that can be deployed with the DERIVA suite.

The interfaces are well-defined and alternative implementations may be developed thus integrating different IdPs (e.g., LDAP, Active Directory, etc.) into DERIVA easily.

[2]https://tools.ietf.org/html/draft-kunze-bagit-13
[3]https://github.com/ruebot/bagit-profiles

V. USE CASES

To validate the utility of a DAMS based approach to scientific data management as well as the applicability of the DERIVA platform, we have applied DERIVA to a range of different eScience use cases. In each situation, the resulting solution was provided to domain scientists who are using it as part of their ongoing scientific explorations. Many of these case studies are in their initial phase and we do not have quantitative data as to the impact of DAMS based approaches. However, from these studies we can conclude that the underlying DERIVA platform can be readily adapted to diverse domains, and that the domain scientists report that these systems are simplifying the process of getting their science accomplished.

Generation of atlases of kidney development. The goal of this collaboration is to collectively annotate high-resolution microscope images so as to trace the development of anatomical structure in the developing human kidney. DERIVA automatically ingests images taken from a microscope and puts them into the asset catalog during which high-quality images are identified based on visual inspection. The domain model for images was extended to include annotation locations, an anatomical term obtained from a controlled vocabulary and a running set of comments on the annotation. Facets are used to manage annotation status. Policy mechanisms are used to separate the ability to annotate, comment, and make the final decision on the annotation value.

Platform for Phenotype Wide Expression (PheWAS) from neuroimaging studies. For a given brain scan, it is possible to computationally produce a large number of phenotypes such as the size, density and curvature of each region of the brain. By aggregating these phenotypes across multiple subjects it is possible to make connections to specific genetic conditions. In practice, domain scientists become rapidly overwhelmed by the tens of thousands of files that contain the phenotype values, by keeping track of image based quality control, and the association of statistical analysis to specific data sets. We are using DERIVA to automatically ingest all of the analysis results and associate them with images, to track necessary manual review of some of the results, and to assemble data sets of phenotype to input to statistical analysis tools to look for significant correlations.

Determination of three dimensional structure of G-Protein Coupled Receptors (GPCR). Protein structure determination by X-Ray diffraction requires many steps to synthesize and analysis steps to crystallize and measure the protein. At each stage measurements are made and analyzed, and at the final step, large amounts of diffraction data must be collected and processed to reveal the protein structure, We are using DERIVA to manage the data that is being generated by a multi-site consortium. Our platform is automatically acquiring and integrating data spanning protein design, flow cytometry, chromotography and gel electrophorsis. Policy mechanisms distinguish between academic and industrial affiliates.

VI. Related Work

Digital asset management systems have been used widely by creative and business organizations, however, the closest comparisons supporting science may be imaging [15] and microscopy management systems [16]. There is a lack of general-purpose asset management capabilities that can span a wide range of multi-domain, multi-modal scientific data and which support the dynamic, rapidly evolving, heterogeneous research activities. Electronic Notebooks such as IPython and Jupyter, are another approach to organizing and visualizing scientific data. However, these approaches do not provide facilities to capture data from instruments. and they are not intended to manage large volumes of data throughout their many transformations.

Current systems and tools to support data-driven discovery, such as computational pipeline systems (e.g. [17]) tend to focus on computation and data analysis. While analysis is clearly important, we assert that data is the currency around which discovery is made and we should take the perspective that discovery is largely data-centric, not process-centric.

Digital repository systems, such as DSpace [18] and Globus Publish [19], may be used to develop institutional repositories which support preservation of digital works and enable open access to data. Digital repositories are primarily concerned with publication, as opposed to the discovery process itself where ones understanding of the domain model may evolve considerably.

Dataspaces [20], and SQLShare [21] also advocate the need for incrementally expanding and evolving data workspaces for individual or collaborative usage. [22] also argue that introspection is a necessity for dynamic, fluid databases, a concept that we further develop in our approach. While these approaches agree with our observations of rapidly evolving, heterogeneous scientific data, they only address query and analysis not capture of assets, organization and annotation.

VII. Conclusions and Future Work

The challenges of data management in eScience applications require a new data-centric approach and we have identified digital asset management as being suitable. We have identified a common set of principles for applying asset management to scientific data and have shown through diverse use cases that a platform built on these principles can benefit a broad range of eScience applications.

In future work, we plan to provide tighter integration of ontologies into the platform. Of particular interest is in defining and refining ontologies as part of the evolution process. Currently, automation is implemented in a combination of cron jobs and shell scripts. We plan to develop a more complete automation service around event-condition-action rules to trigger automation activities and to allow user interfaces to specify automation activities on given events. We also plan to provide more user friendly tools for dynamically evolving the data model. We will need to introduce model versioning along with value versioning as part of this work. Finally, we are integrating online analytics into the system, with the goal of allowing the presentation and interaction to adapt to common usage patterns.

Acknowledgments

The authors wish to thank the Deriva developers: Jennifer Chen, Joshua Chudy, Mike D'Arcy, Laura Pearlman, Mei-Hui Su, Jessie Wong, and Serban Voinea. Alejandro Bugacov, Anoop Kumar, Hongsuda Tangmunarunkit, and Cristina Williams have contributed significantly to use case analysis and deployments. We also thank our science collaborators, Andrew McMahon, Jill McMahon, Seth Ruffins, Michael Hanson, Raymond Stevens, Scott Frasier, Donald Arnold, and William Dempsey.

References

[1] P. Fox and J. Hendler, "The Science of Data Science," *Big Data*, vol. 2, no. 2, pp. 68–70, jun 2014.

[2] C. Goble *et al.*, "Accelerating Scientists' Knowledge Turns," in *Communications in Computer and Information Science*, 2013.

[3] B. L. Claus and D. J. Underwood, "Discovery informatics: its evolving role in drug discovery," *Drug Discovery Today*, vol. 7, no. 18, pp. 957–966, sep 2002.

[4] S. Kandel *et al.*, "Enterprise data analysis and visualization: An interview study," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 18, pp. 2917–2926, 2012.

[5] R. G. Steen *et al.*, "Why Has the Number of Scientific Retractions Increased?" *PLoS ONE*, vol. 8, no. 7, 2013.

[6] C. G. Begley, "Six red flags for suspect work." *Nature*, vol. 497, no. 7450, pp. 433–4, may 2013.

[7] J. Gray *et al.*, "Scientific data management in the coming decade," *SIGMOD Rec.*, vol. 34, no. 4, pp. 34–41, 2005.

[8] I. M. J. C. R. Licklider, "In Memoriam: J. C. R. Licklider 1915-1990," 1990.

[9] T. Clark *et al.*, "Micropublications: a semantic model for claims, evidence, arguments and annotations in biomedical communications," *Journal of Biomedical Semantics*, vol. 5, no. 1, p. 28, 2014.

[10] B. Plale *et al.*, "SEAD Virtual Archive: Building a Federation of Institutional Repositories for Long-Term Data Preservation in Sustainability Science," *International Journal of Digital Curation*, vol. 8, no. 2, pp. 172–180, nov 2013. [Online]. Available: http://ijdc.net/index.php/ijdc/article/view/8.2.172

[11] B. Howe *et al.*, "Database-as-a-Service for Long-Tail Science," Portland, Oregon, 2011.

[12] S. Jain *et al.*, "SQLShare: Results from a Multi-Year SQL-as-a-Service Experiment," in *SIGMOD'16*. San Francisco, CA, USA: ACM, 2016.

[13] L. Moreau, "The Foundations for Provenance on the Web," *Foundations and Trends® in Web Science*, vol. 2, no. 2-3, pp. 99–241, 2010.

[14] S. Bechhofer *et al.*, "Why linked data is not enough for scientists," *Future Generation Computer Systems*, vol. 29, no. 2, pp. 599–611, 2013.

[15] D. S. Marcus *et al.*, "The Extensible Neuroimaging Archive Toolkit: an informatics platform for managing, exploring, and sharing neuroimaging data." *Neuroinformatics*, vol. 5, no. 1, pp. 11–34, 2007.

[16] J. R. Swedlow *et al.*, "Bioimage informatics for experimental biology," *Annual review of biophysics*, vol. 38, pp. 327–46, jan 2009.

[17] J. Goecks *et al.*, "Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences." *Genome biology*, vol. 11, no. 8, p. R86, 2010.

[18] M. Smith *et al.*, "DSpace: An Open Source Dynamic Digital Repository," *D-Lib Magazine*, vol. 9, no. 1, 2003.

[19] K. Chard *et al.*, "Globus data publication as a service: Lowering barriers to reproducible science," in *11th IEEE International Conference on eScience*, 2015.

[20] M. Franklin *et al.*, "From Databases to Dataspaces: A New Abstraction for Information Management," 2005.

[21] B. Howe *et al.*, "Automatic example queries for ad hoc databases," in *Proceedings of the 2011 international conference on Management of data - SIGMOD '11*. New York, New York, USA: ACM Press, jun 2011, p. 1319.

[22] A. Halevy *et al.*, "Principles of Dataspace Systems," Chicago, Illinois, USA, 2006.