Analytic Technology Industry Roundtable

Roundtable Analysis Exchange Model 1.0 Software Design

The following document describes the Roundtable's *Analysis Exchange Model 1.0*. The document captures the entire software design (system architecture), including ontology, knowledge store, and software for transfer and adaptation. We also describe in detail some planned features for the next iteration of the model, such as a publish and subscribe service. This document also lays out an example use case and test data, which helped to mature the architecture.

The Analytic Technology Industry Roundtable is investigating avenues for collaborative activity between data and analytic capability providers. The Analysis Exchange Model 1.0 facilitates this collaboration with a client-server architecture for storing shareable results, an ontology spanning multiple domains of interest, software for both parsing and adapting analytic results, and a query engine for retrieval.

Approved for Public Release; Distribution Unlimited. Case Number 17-2811

1. Background

1.1. Toward the Analysis Exchange

The Analytic Technology Industry Roundtable produced several studies focusing on analytic architectures to gain a better understanding of the opportunity for collaboration [1, 2, 3, 4]. These studies cover surveys of government and industry architectures, analytic capabilities, and use cases. They provided the foundation for the scope and goals of the Analysis Exchange Model presented in this document.

The Analysis Exchange Model 1.0[1] architecture offers the capacity to publish and subscribe to shareable analytic results, using a common ontology as a shared vocabulary of the knowledge each of the disparate analytics produce. This work fosters the adaptation of analytic results into this common representation.

A reference implementation of this design work is underway, following use cases agreed to by government and industry members of the Roundtable. The core architectural components of the Analysis Exchange Model 1.0 support the desired collaboration, which includes both the planned components and the essential ones that will be available as part of the release of the first reference implementation. The exchange design is based upon eighteen to twenty-four months of engagement, study, and work within the Industry Roundtable Working Group.

1.2. Related Work

The Analysis Exchange Model 1.0 principally intends to support collaboration and knowledge exchange between industry partners who provide data and analysis, allowing different sources to create an enriched final analysis product. In a broader context, there are other similar efforts that deserve mention.

Several of the concepts applied in the Analysis Exchange Model were drawn from prior work on the MOSAIC [5, 6] project, which also sought to achieve an exchange of analytic output from varied solutions. In MOSAIC's

[1] In our original material, this was referred to as the *Common Exchange Service*. The current MITRE implementation of the Analysis Exchange is codenamed Tangerine.

context, these solutions were independently developed new and legacy analytic software bridged through a common data representation, but loosely coupled to allow for replaceability.

Another significant and relevant effort is the National Information Exchange Model (NIEM).[2] NIEM is a mature initiative spanning government agencies that supports information sharing and data exchange and offers building blocks to achieve this.[3] These building blocks include reference schemas for core and domain-specific elements and adaptation capabilities for certain existing standards. NIEM's emphasis has been on interagency exchange as opposed to directly supporting industry collaboration as the Analysis Exchange seeks to do. Nevertheless, being a general platform for the exchange of information between organizations, NIEM is a possible platform for implementing an Analysis Exchange, which is a specific platform for the exchange of analytic results and their source information.

[2] https://www.niem.gov/techhub/iepd-resources
[3] https://release.niem.gov/niem/4.0/

2. Overview

2.1. Principal Components

The Analysis Exchange Model 1.0 (as depicted in Figure 1) describes the *Analysis Exchange*, a hub that brokers analytic artifacts, knowledge, and results between analytic stacks and the analyst customers that make use of them. The architecture's principal components include the *Collaboration Services*, a *Transfer Service*, a *Knowledge Store*, and tacitly any external *analytic stacks*. The analytic stacks are external to the architecture and can operate

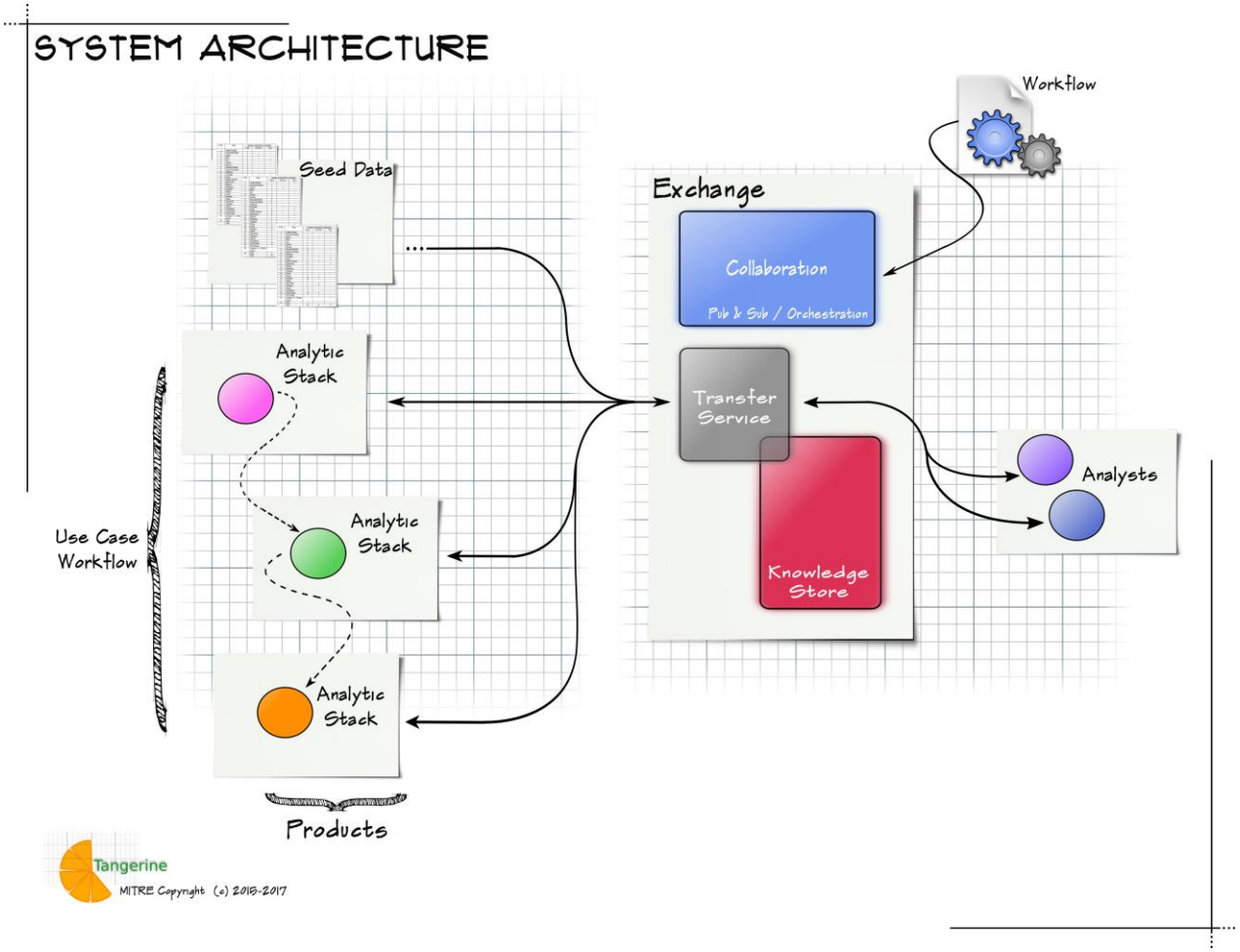

Figure 1: Exchange for automated analytic collaboration across industry and government.

independently of one another, only interacting when they have permission and need for results outside their own environment. The other principal components of the architecture ensure the Analysis Exchange can support that activity.

The **Collaboration Services** are the components that drive and manage the workflow as analytic stacks interact with the Analysis Exchange. While not a part of Analysis Exchange Model 1.0, this will eventually include a *Pub/Sub Service*, which allows the publication of and subscription to analytic results. This can be either as part of workflows or applied in an ad hoc manner.

The **Transfer Service** is responsible for the bulk or streaming transfer of results between the Analysis Exchange architecture and external providers or consumers. It can be divided into the transport subcomponents and the adaptation subcomponents, as shown in Figure 2. Because there is not a tight integration mandated or expected between external analytic stacks' products, a key part of the Transfer Service are **Adapters**. An adapter is software that converts results between a variety of native analytic models and a common data model with a supporting ontology that has been defined for the Analysis Exchange. This adaptation allows for usability across varied independently developed and executing analytic stacks, and it is a crucial element of the Analysis Exchange.

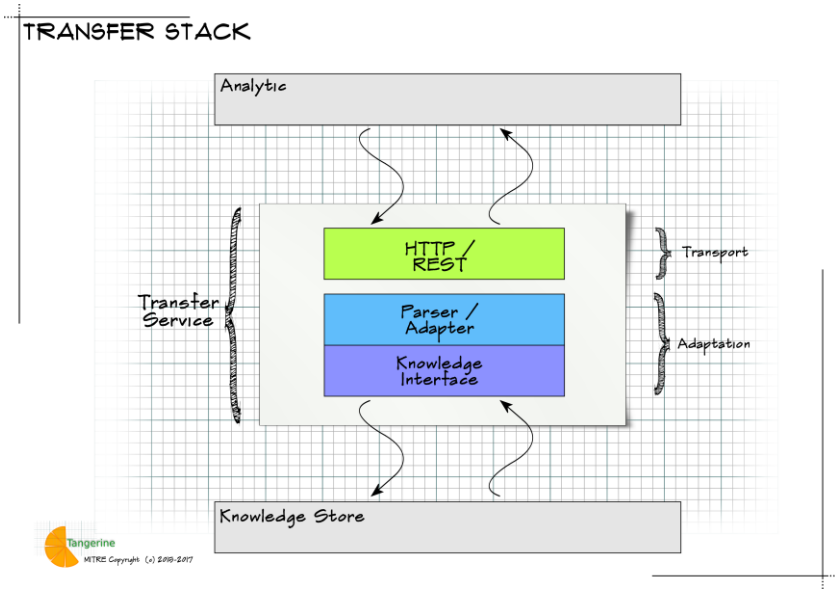

Figure 2: Transfer stack

The **Knowledge Store** is where results from the analytic stacks that have been passed through an adapter are held. The results' retention can be temporary for results that are merely kept in interim between for a workflow instance involving multiple analytic stacks with data dependencies on one another. The retention can also be persistent, where more significant results or data can be kept and used across multiple workflow instances, which is to say an actual run of

a planned workflow of analytic stacks interacting with the Analysis Exchange.

2.2. Goals

The overarching intent of the Analysis Exchange is to achieve fusion of the artifacts and knowledge it holds so different industry automated analytics can contribute to a richer final product. Of note with this design is the attempt to reduce the direct integration dependency between analytics stacks by following a loosely-coupled model. Further analysis of what the Analysis Exchange holds is possible downstream of this service.

From the outset, there were specific goals identified as being necessary for the Analysis Exchange to be successful:

- Providing an architecture to support collaboration, creating the fundamental access methods for the Analysis Exchange while allowing the contributing analytic stacks to maintain independence from the architecture.
- Creating the *Analysis Exchange Ontology*, a data model with the supporting ontology that can be publicly released and serve as a lingua franca for analytic results (this ontology described in more detail below) held in the Knowledge Store.
- Generating the templates, or at least the pattern of activity, to allow for the adaptation of different industry analytic results (via an adapter) to align to the Analysis Exchange Ontology.

These goals introduce the different components developed for Analysis Exchange Model 1.0. Some are developed from scratch and others by applying and extending existing capabilities. These include the following:

- The ontology serving as the underlying data model and rules for reasoning.
- A semi-persistent store for the analytic results mapped into the Analysis Exchange Ontology.
- The application programming interfaces (APIs) to allow for interaction with the Analysis Exchange.
- Template parsers and adapters for migrating analytic results into the Analysis Exchange Ontology.

Later sections offer more details on these components, but first we summarize how these components fit together. The internal Knowledge Store retains artifacts for a duration the use case requires. Further, the results in this store are defined by a consistent data model that conforms to an ontology built for the supported use case (Analysis Exchange Ontology), which seeks to align the semantics of external analytic stacks' results and mitigate ambiguities between different representations. The APIs (part of the Transfer Service depicted in Figure 1) are the entry point into the architecture. In later iterations, these can be controlled following an ad hoc publish-

subscribe pattern and can also be extended to allow workflow orchestration to call on external analytics—and potentially internal analytics eventually; these eventual components fit under Collaboration Services. The adapters are required to make the format and possible semantic conversion into this model. These adapters ensure that downstream analysts receive a consistent picture of the results regardless of the source.

3. Architectural Components

While Figure 1 depicts the layout of the key architectural components and their relationship with analytic stacks, Figure 3 lays out the implementation level picture of these components, both elements planned for Analysis Exchange Model 1.0 and those that are proposed. This section explores each component's function and the specific implementation choices for Analysis Exchange Model 1.0. These components include the Analysis

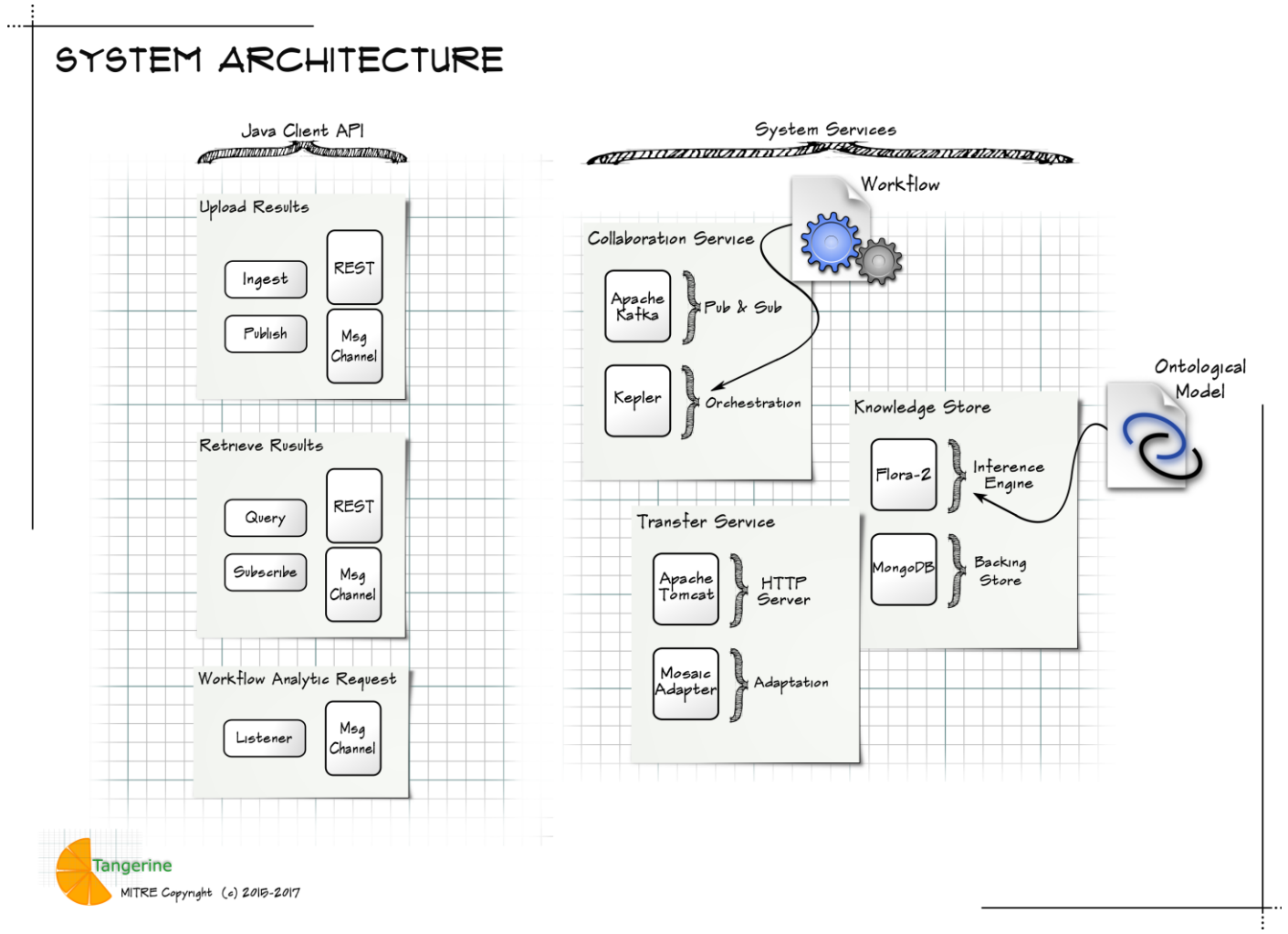

Figure 3: Implementation level diagram of the Analysis Exchange

Exchange Ontology, the Knowledge Store including the inference and query engine and its backing store, and the Transfer Service, which includes the externally facing APIs and, importantly, the adapters. Some of the key planned architectural components are also described, such as the Pub/Sub Service.

3.1. Analysis Exchange Ontology

3.1.1. Analysis Exchange Ontology Design

An underlying need in a successful Analysis Exchange is achieving a seamless integration across varied analytic results in support of a use case. These results are often in semantic models or can be made to conform to them. We envision applying an ontology (namely a set of defined classes, relationships, and rules) that describe the elements relevant to the use case, which allows for a consistent treatment of the knowledge and the capacity for reasoning across the rules that describe how different statements of knowledge affect one another.

The goal of a comprehensive ontology that can cover all use-cases is quite lofty. So far, no ontology has sufficed to accomplish this, and the Analysis Exchange Ontology is not an exception. Instead, it intends to be as extensible as possible to new use cases, providing each complete and accurate coverage of entities and relationships relevant to it. Therefore, the most important element of the approach is to build the ontology such that it can handle diverse, unrelated domains and handle collisions that may occur due to incompatibilities or ambiguities across domains.

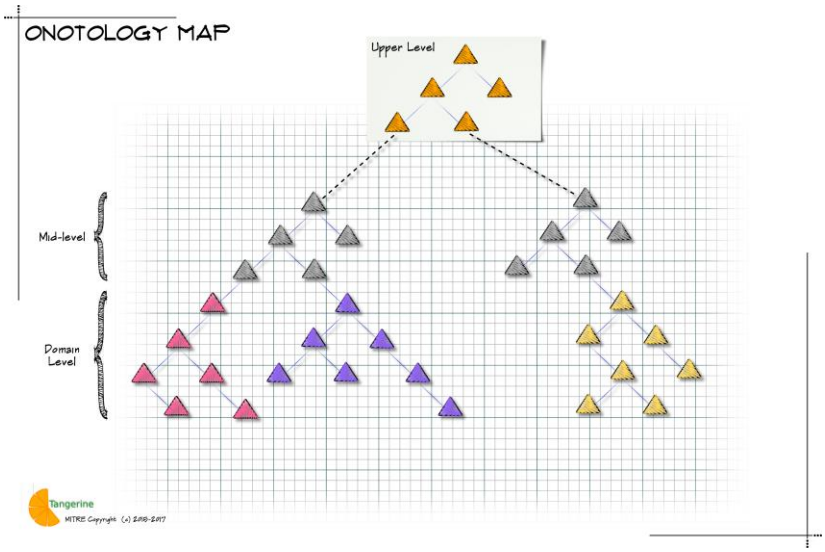

Figure 4: Ontology structure

The ontology here follows a typical division of ontological levels into upper, middle, and lower that aid in expressing the scope of the elements, as Figure 4 suggests. Elements

that are universal across all knowledge domains are in the upper ontology, a popular example of an extensible upper ontology being Basic Formal Ontology (BFO) [7]. BFO is a small relative to many other ontologies, but meant for extension into multiple domains representing analysis output. It lacks terms for specific sciences, but it has a framework that has been extended into approximately 250 ontological efforts currently because it offers a formal method for creating complete domains. This notion of extensibility is also key to our approach. Further development of the Analysis Exchange Ontology depends on the use cases to be supported. These use cases must be clear and be well understood to ensure the knowledge essential to achieving their goals is captured in the ontology's classes, relationships, and rules.

At the middle and lower levels, the ontology becomes more specific to a set of domains (middle) and then to a specific domain (lower). Many of the terms that appear in the Suggested Upper Merged Ontology (SUMO) [8] span the upper level down into the middle level per this definition, and our ontology draws on SUMO's hierarchy, which has a wide coverage, to flesh out much of its structure. The elements that are more particular to a given use case and its domain are added to the ontology to ensure that every entity, relationship, and property of relevance to the use case appears.

Because of this structure, the ontology naturally can be quite complex, but many of the elements in the upper ontology that provide a backbone for representation rarely are instantiated or need to be known outside the ontology development; they serve to provide a behind-the-scenes structure. Rather, it is more common that specific entity types, relationships, and properties that are defined at the middle and lower level are those which analytic adapters will be mapping.

3.1.2. Analysis Exchange Ontology Implementation

Developing this ontology, or the Analysis Exchange Ontology as described above, is one of the heftiest Roundtable efforts for Analysis Exchange Model 1.0. The ontology is represented in an OWL format that defines the entities, relationships, properties, type hierarchy, and inference rules. In the Analysis Exchange, statements and rules conforming to this ontology are created using Flora-2 based F-Logic, which has proven a sturdy ontology language for similar past data modeling work that involves integrating across different analytic results,

given its a rich expressiveness. This ontology is integrated with the Knowledge Store that is populated with instances of knowledge by the adapters. The store and adapters are both described in later sections.

3.2. Knowledge Store

3.2.1. Knowledge Store Design

While analytic artifacts published by the Analysis Exchange conform to the Analysis Exchange Ontology, they also need a place to be stored internally. Depending on the use case, this Knowledge Store can be permanent, volatile, or somewhere in between. This design intends to support any of those needs. In some cases, all that is of interest is timely data whereas in others, aggregation over successive uploads provides the needed picture to downstream analytics and analysts. There can also be degrees of volatility, where all knowledge is at least retained temporarily and some is always retained long term. Regardless, this suggests a knowledge base that can potentially scale if the analytic stacks produce considerable artifacts.

3.2.2. Knowledge Store Implementation

In Analysis Exchange Model 1.0, MongoDB is the basis for the store, and it retains the knowledge artifacts represented in F-Logic. Prior work provides a basis for this synthesis of MongoDB and F-Logic, which makes it clear that this is a feasible solution; we leverage that experience. The length of retention is determined by what fits a given use case.

The Knowledge Store's organizational convention is a table (or collection as represented in MongoDB) per ingested document, where each document represents an independent instance of contextually connected entities. When further results enrich this, they become new tables that have the following naming scheme: *XX_UUID_EPOCH*. In this scheme, *XX* is a consistent abbreviation of the analytic stack that produced it (or IN for ingest), *UUID* is a common universally unique identifier, and *EPOCH* is the creation time for the table.

Further, when instances are declared, they can follow any number of naming conventions if different instances are given unique symbols. The recommendation when starting out is to follow a naming convention using the table's name. Namely they are structured as *XX_UUID_EPOCH_#*, where the first three elements match the table

and # is a temporary incremented number to indicate the order in which they were produced. This is not intended to serve as provenance, but is employed to make the results more easily traced for debugging. When adaptation is verified to work for initial test cases, this can be simplified to just using different UUIDs for each instance symbol, which will be considerably more space efficient.

Note that this scheme assumes that tables will be produced and added to once. All new results will be added to new tables. MongoDB can retrieve interrelated statements that cross multiple tables (or collections), but it does require keeping track of the tables being created during a workflow instance or remembering the names of tables if results are intended to be used later.

3.3. Transfer Service and Adapters

The Transfer Service consists of both transport and adaptation activities. This section expands on these activities by examining the APIs and the adapter software.

3.3.1. API Design

The principal goal of the Analysis Exchange is to bridge capabilities between different industry analytic stacks in service of government customers and analysts. The APIs for the Analysis Exchange are how external analytic stacks and end customers will access the Analysis Exchange. They cover activities such as uploading results, collecting results to ingest in downstream automated analytics, and retrieving the results that represent the outcome of a complete activity, whether as an instance of a repeatable workflow or a single ad hoc experiment bridging multiple analytic capabilities. These APIs can be the foundation of providing a Pub/Sub Service or more robust centralized orchestration that calls on external analytics to execute. Eventually, there can also be a capability in the APIs to allow for analytic providers and customers to execute internal analytic processes in the Analysis Exchange, but the initial vision for Analysis Exchange Model 1.0 expects the analytics to remain external.

3.3.2. API Implementation

For Analysis Exchange Model 1.0, these APIs are written in Java. The Analysis Exchange uses RESTful APIs, or web service APIs, and employs a standing Apache Tomcat server.

3.3.3. Adapter Design Basics

One of the more serious challenges to inter-industry and government collaboration is the disconnect in formats and models of analytic results. For analytics to operate in concert, whether they ingest one another's results as input or synthesize a final product collectively, they must have a common language. This is the purpose of the Analysis Exchange Ontology, but results must be mapped into (and in some cases out of) the Analysis Exchange Ontology, as we know that we cannot mandate that every analytic will produce (or consume) this data model. This is where adapters become essential. Adapters are derived from a concept articulated in the MOSAIC project [6, 5], which dealt with combining artifacts of loosely coupled analytics that did not natively communicate.

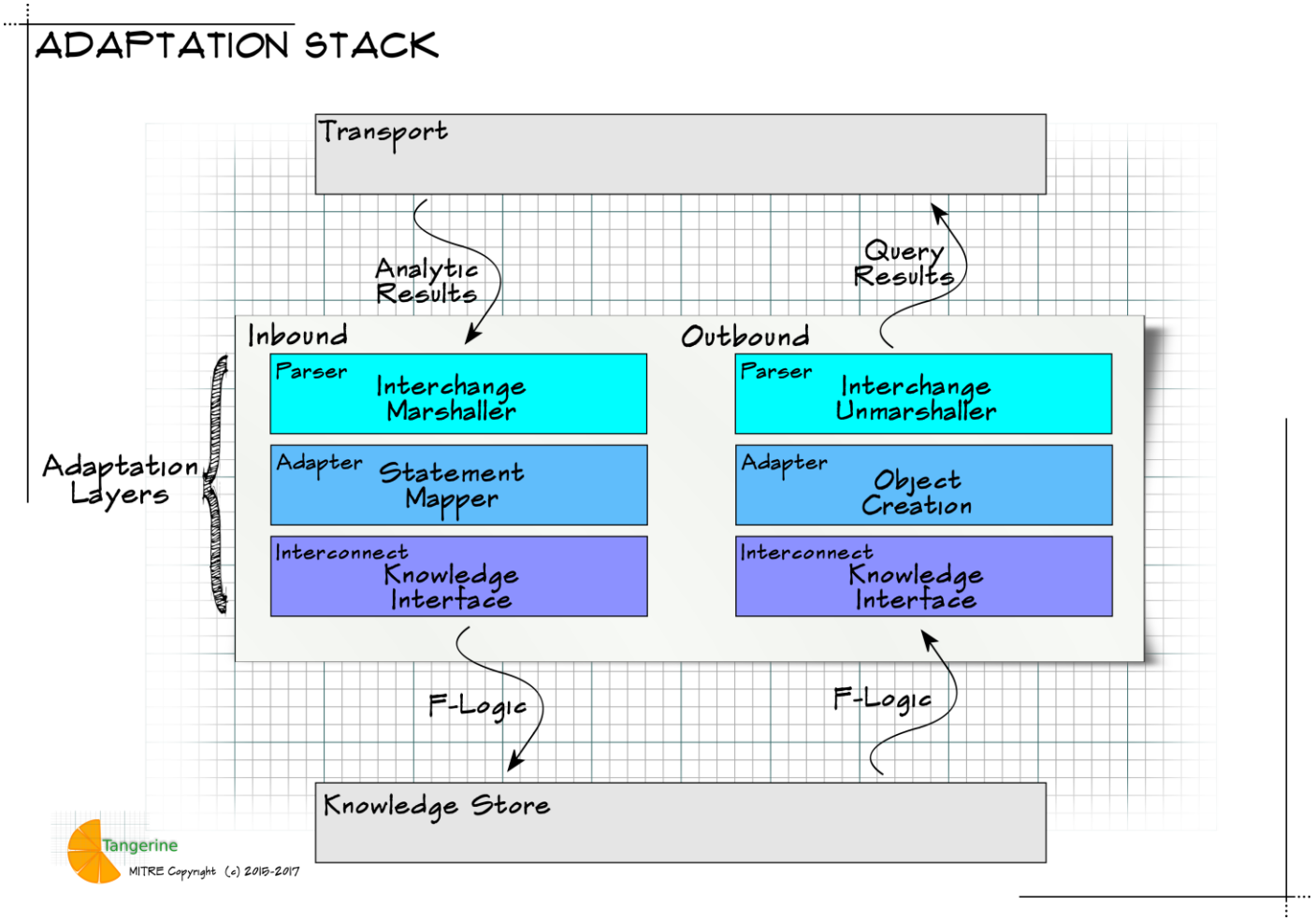

Figure 5: Adapter stack

Adapters are the software that converts the results from their raw analytic results into a format conforming to the Analysis Exchange Ontology and vice versa if the use case dictates. This also means that there must be adapters for each analytic involved or at least each analytic result type involved; formats can potentially span different analytics.

Figure 5 depicts the adapter stack, and the typical operations that occur both when adapting content into the Analysis Exchange Ontology and when adapting content back into a form that external analytics expect. This exists between the transport layer and the store in the overall stack for the Analysis Exchange. For the inbound adapter, raw input or native analytic results are marshalled into JSON by the parser. The content is then semantically altered as it is mapped into statements conforming to the Analysis Exchange Ontology by the adapter. The interconnect moves the knowledge into the store. The outbound adapter essentially reverses this process, where the knowledge statements are queried and then adapted into JSON records. This is then unmarshalled into any format designed to be sent across the transport to the downstream analytic.

3.3.4. Adapter Design Scope

Knowing how to scope adapters for an analytic applied to a use case requires ongoing collaboration to ensure the analytic output is understood and the adaptation is preserving its semantics. This semantic alignment is a difficult problem and should not be underestimated, as mistakes or shortfalls in representation can lead to propagating errors and rendering the resulting knowledge as untrustworthy. Careful creation and curation of the semantic alignment is therefore one of the more intensive parts of applying the Analysis Exchange Ontology to new use cases, but it also focuses much of the required effort into specific software.

Because the ontology behind the Analysis Exchange is complex and is intended to span different use cases, it can easily appear daunting to someone who only wishes to use the Analysis Exchange in a single use case. From the perspective of many analytics, what they provide and require is much narrower than the scope of the ontology, but even a suite of analytics from different sources with a seemingly comprehensive use case will not span the entire ontology. Certainly, ontological elements that provide the upper infrastructure of the type tree fall into the category of those that should be invisible at the use case level. Typically, it is only the entities and relationships

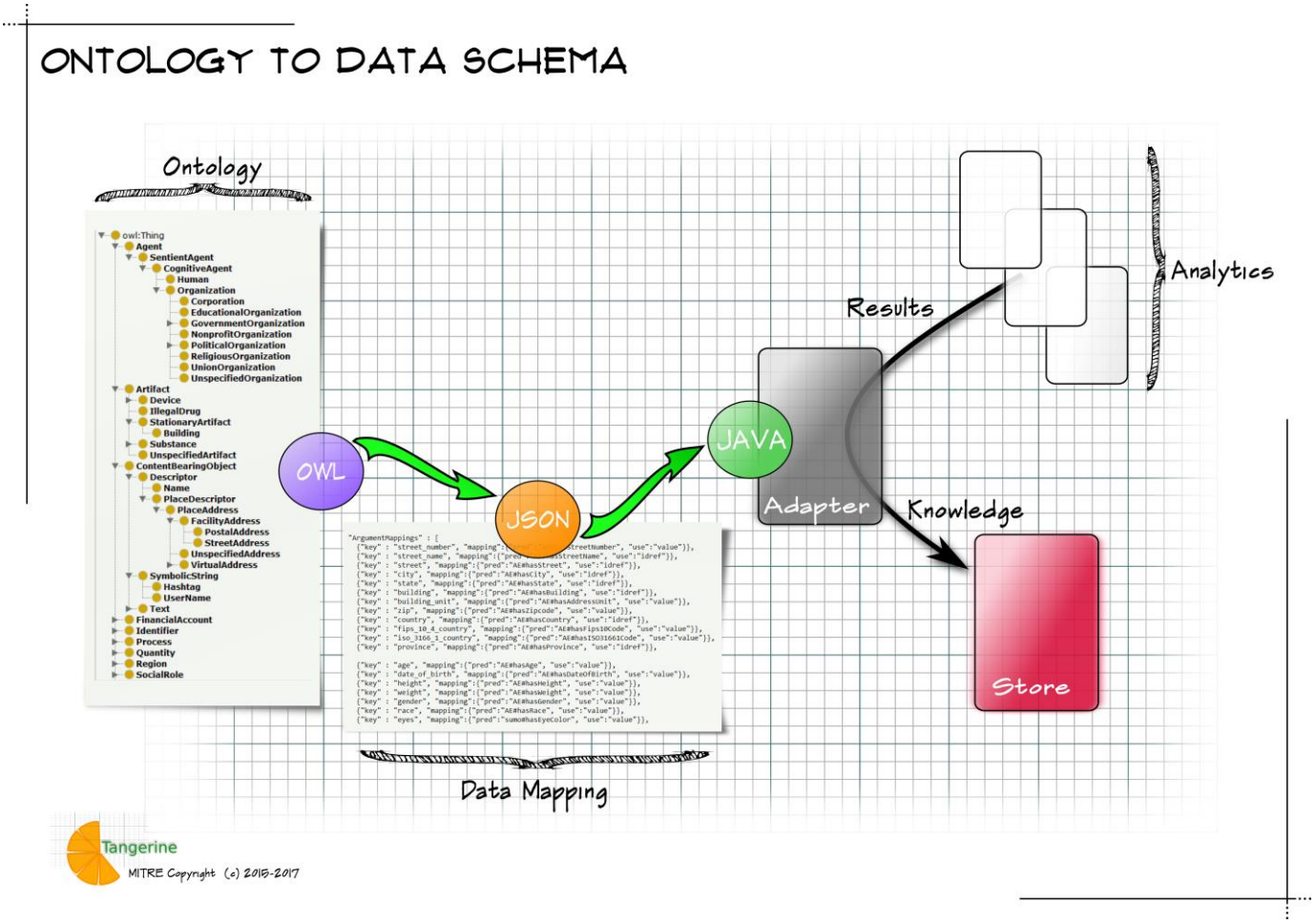

Figure 6: Relationship between ontology, mappings, and the adapter.

that fall within a use case's domain that are needed. The use of the adapters aids in limiting the perspective to only the elements of the overall ontology that are relevant for a specific analytic stack's contribution to a use case.

As shown in Figure 6, the overarching ontology is used for a given use case and analytic to guide the creation of a data mapping from the source schema and terminology to those that conform to the Analysis Exchange Ontology's semantics. This mapping is then used by the adapters to represent the entities, relationships and properties that exist within the analytic output. The conversion conforms to the mapping, sidestepping results that are irrelevant to the use case and are not adapted and ignoring completely most of the ontology that is not immediately applicable to what is specifically being represented in the output. In future iterations, the ontology

can provide guidance for changes to the data mapping (and thus the adapter) and it can also offer rules that infer over the statements provided by the analytic, which may evolve over time.

3.3.5. Adapter Options for Adding and Merging Knowledge

Diving deeper into the adaptation process, the inbound flow has certain options that will be meaningful when analytic stack results are enriching the knowledge for a specific workflow instance. These include the following:

1. Add original analytic stack results to a new table.
2. Add original analytic stack results to an existing table.
3. Add resulting knowledge statements to a new table.
4. Add resulting knowledge statements to an existing table.
5. Connect the resulting knowledge statements to entities in an existing table where there is a consistent possible connection point.
6. Connect the resulting knowledge statements to entities in an existing table, further defining where the connections occur as there are multiple possible connection points.

The first two options are the most straightforward, as results have been created by an analytic stack and added directly to the Knowledge Store *without* adaptation. This makes sense for use cases where an analytic stack's results can be used downstream by another analytic stack without requiring modification or needing adaptation into the Analysis Exchange Ontology. This assumes the downstream analytic stack can natively ingest the upstram analytic stack's results.

Options 3 and 4 are slightly more involved than the first two options, as results have been created by an analytic stack and then adapted into the Analysis Exchange Ontology. In this case, it is assumed any downstream analytic stacks cannot or will not use the native output of the analytic stack producing these results. These two options make an important assumption: that the new statements are independent of the instances already in the table. This means that they will not have references to any existing instance and therefore their statements can be added as is.

Option 5 is more complicated. In this case, statements refer to one or more existing instances in the table (e.g., newly discovered properties about a person already present in the Knowledge Store). The graph of new statements must therefore be linked to the graph already present in the table. In the adapted, these connection points must be provided when invoking the adapter. A connection point refers to an instance already stated in the table's graph. Relationships and properties in the adapter will feature this as one of their arguments, either the subject or object depending on the knowledge structure. This requires the instance symbol to be passed along with the adapter so it can properly create the new statements. Note, there is an underlying assumption that the analytic stack results would only have one, unambiguous connection point to map to the existing instance in the table.

Option 6 is somewhat similar to option 5. This eliminates the assumption discussed for option 5, and there are multiple possible connection points in the results which could map to the existing instance in the table. In addition to providing a connection point symbol, it is also important to indicate what element in the native analytic stack results matches this connection point symbol. With the elements selected both from the source and the destination, the new statements generated can be added to a table's existing graph.

3.3.6. Adapter Implementation

In Analysis Exchange Model 1.0 and as Figure 6 describes, the adapters are developed in Java, the data mappings that define an adapter's behavior are composed in JSON, and the overarching ontology is created in OWL format.

While adapters can be bi-directional (going from raw analytic results to a format conforming to the Analysis Exchange Ontology and from the Analysis Exchange Ontology to a format ingestible by external analytics), we start with adapters that map from analytic results into the Analysis Exchange Ontology and create adapters in the other direction as required by the use case, so that the adapters necessary to execute and demonstrate the use case are created. These adapters can eventually become the basis for the continual development of more adapters for other analytic results.

3.4. Pub/Sub Service

3.4.1. Pub/Sub Service Plans

Following a publish-subscribe pattern for communication is a straightforward method that imposes a minimal burden. This model expects that when analytic artifacts are made available, the Analysis Exchange will publish these artifacts. Therefore, subscribers awaiting certain expected analytic artifacts will be notified when the artifacts that interest them, either for a single experiment or part of a regular workflow, are available.

Analysis Exchange Model 1.0 will not include a Pub/Sub Service, but it will be expandable to include this. In future implementations, the Pub/Sub Service is planned to be crafted using Apache Kafka and integrated with the APIs described earlier. While a publish-subscribe pattern typically assumes an underlying structure of clients and a central server, Kafka is used to change this structure into one that is peer-to-peer. This allows it to act as a messaging service between the Analysis Exchange and the external systems when updates occur. External analytic stacks with data dependencies or end customers will subscribe to the artifacts published by the Analysis Exchange. Furthermore, these analytic stacks provide the artifacts that the Analysis Exchange makes available once converted to conform to the Analysis Exchange Ontology. Who acts in what capacity should be agreed upon as part of defining an initial use case. This also requires developing a listener that ensures events are noticed, where events are any kind of relevant notification about a change in the system or data or any requests that are part of a workflow.

Eventually, this Pub/Sub Service can provide the basis for more complex internal orchestration, where a centralized control calls on both external and internal analytics to execute on different stages of artifacts to produce the final-state results. The initial vision intends to keep this simple, but with the potential to be extended to greater responsibilities held by the Analysis Exchange, perhaps using Kepler, an open source scientific workflow application, as the basis for the orchestration. The listener we described in the publish-subscribe pattern could also be used in this more central orchestration.

4. Architectural Organization

4.1. Layout

The architecture and its components described in the previous sections conform to an underlying system stack, as shown in Figure 7. The pattern followed in the Analysis Exchange Model 1.0 (and captured here) is a client-server model. This allows external facing clients to expose an interface for analytics to provide (or query) results stored in the Analysis Exchange, while a central server is responsible for storage, whether temporary or long-term. Examining this in greater detail, the Analysis Exchange's Java APIs are the application interface that all

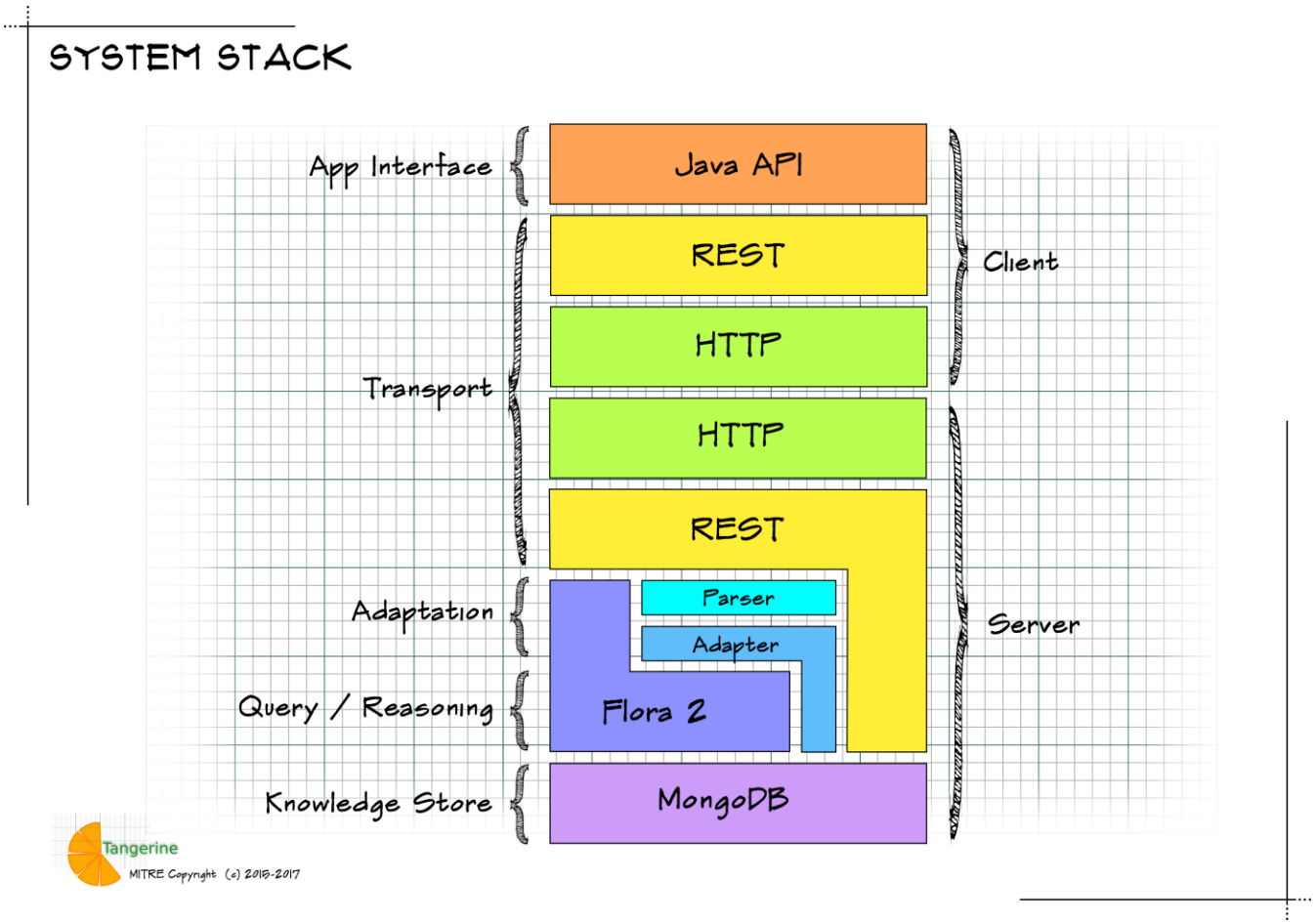

Figure 7: System stack for the Analysis Exchange

external analytics will use. Input and queries are made through the REST protocol and HTTP between the client and the central server to transport the information moving between them. For input, the parser and adapter software handles changes to the format and semantics of the data, changing both so they fit the data store and the ontology's expectations for the common representation in the Analysis Exchange Ontology.

For query, a query and reasoning engine built as a modified version of Flora-2 offers the method for pulling from the store. The store, where knowledge adapted from analytic results and ingested data is kept, is implemented as MongoDB storing F-Logic statements in JSON. This pattern generally holds for any analytics, making the process repeatable for new or changing analytics that are maintained externally to the Analysis Exchange.

4.2. Flow

4.2.1. Current Flow Implementation

Having established the Analysis Exchange's components and their layout, an examination of the flow of activity through the Analysis Exchange follows. At the heart of activity in the Analysis Exchange is its interaction with external analytics. This is explored both at the level of typical commands through the Analysis Exchange's API and at the level of a larger workflow of multiple analytics.

Figure 8 depicts flow between client and the Analysis Exchange server, which is a web application that resides within a J2EE servlet container. The Analysis Exchange server is shown as the Tomcat server process to the right. Four major types of Analysis Exchange and analytic interaction are depicted. These include commands through the Analysis Exchange's API to submit results into the Exchange's store, query for previous results, fetch results in a specific format, and remove results from the Analysis Exchange.

For a submit activity, commands go through the REST API, which passes the command and the submitted data to the standing Tomcat server. Assuming the results are not already in JSON statements conforming to the Analysis Exchange Ontology, the submission is parsed and, following a data mapping for that analytic, adapted into the F-Logic JSON statements, which are stored in the MongoDB server. A confirmation is sent back through the Tomcat Server and API.

For a query activity, commands go through the REST API, which passes the command and query to the standing Tomcat server. The query is then passed to the Flora-2 based reasoning engine to call the MongoDB server for all statements that match the query. These statements are returned to the Tomcat server and the API to the recipient who made the query.

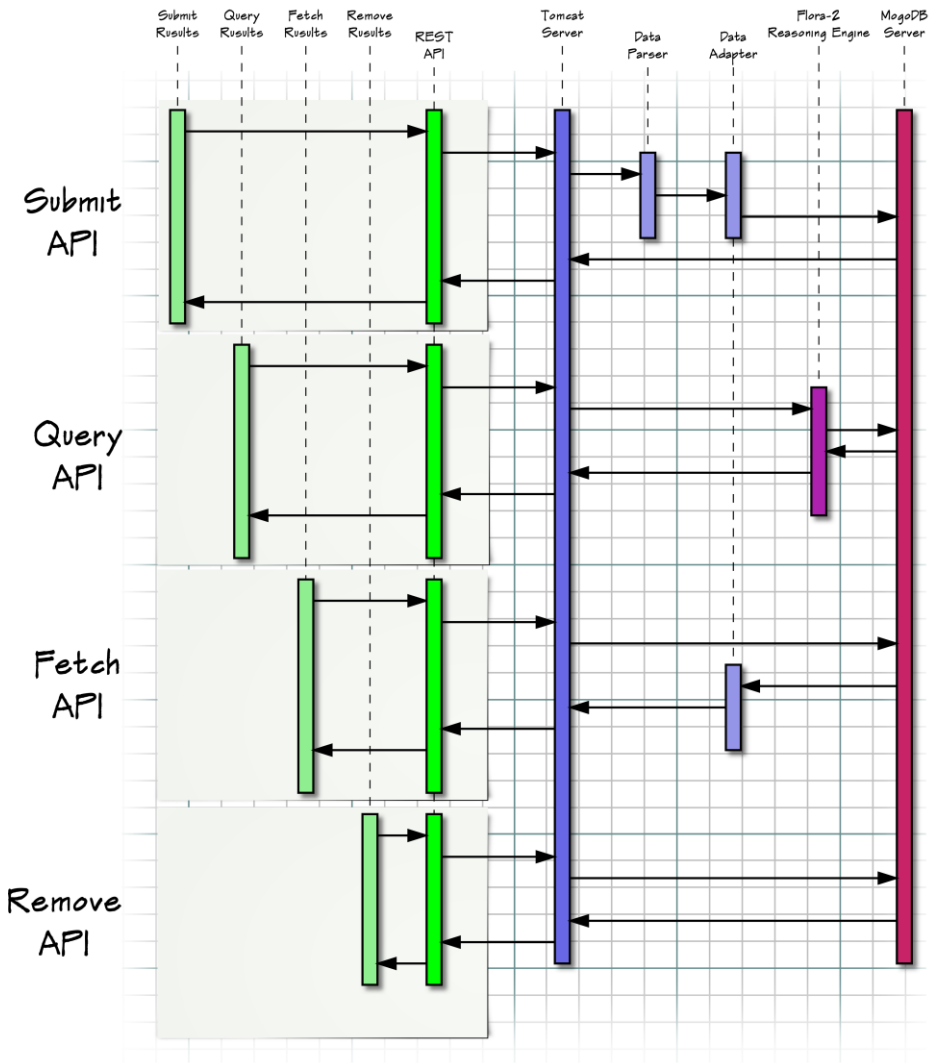

Figure 8: Analysis Exchange API flow

Fetch activity resembles[4] the intent of the query as it draws results previously uploaded to the Analysis Exchange. The key difference is that while the query returns F-Logic JSON statements, the fetch will return these statements adapted into a different format (for example, a CSV with a record per row and a field per column) and different semantics (i.e., mapping the terminology of entities and relationships into terms and fields the fetching analytic understands). Thus, it has a data adapter that will map the statements appropriately. This also only works as a bulk fetch, which means that it will return an entire table of results from MongoDB; the Flora-2 based reasoner is not part of this flow, which goes directly from Tomcat to MongoDB.

The final activity, removal of results, deletes a set of statements. As with the other example, the command goes through the REST API to the Tomcat server. This submits the deletion request to MongoDB, which removes all the qualifying statements from its store. A confirmation is sent back through Tomcat and the REST API back to the application that submitted the command.

4.2.2. Future Flow Implementation

When the Collaboration Services are added in subsequent iterations of the Analysis Exchange, performing integrated workflow in the Analysis Exchange becomes possible. Figure 9 depicts a higher scale workflow of analytics and provides an example of an analytic run, which includes a query for data from the Analysis Exchange and provides new results back to the Analysis Exchange. In this example, analysts kick off a workflow of three analytic stacks (A, B, and C). This goes through the Kafka client to set up a listener for the workflow, which will eventually notify the analyst of when the workflow is complete. Listeners are also then set up for each of the analytic stacks through Kafka clients. Once the analytic stacks have listeners, the Kepler workflow will begin to invoke the analytics, starting with analytic A. The listener recognizes that a command has been issued to tell analytic A's shell to stand up and execute.

In the example (using analytic A), the analytic queries the Analysis Exchange for data before execution and will return its results to the Analysis Exchange. This query follows the same pattern as described above for the query

[4]

activity, invoking the Flora-2 based engine to retrieve the relevant JSON encoded F-Logic statements to return to the analytic. After these statements are passed to analytic A's shell, the analytic performs its execution using these statements. When this completes, the analytic submits its results to the Analysis Exchange, which follows the pattern of the submit activity described above. As part of the workflow, once the parsing and adaption and upload

to MongoDB are complete, the confirmations go back through Kepler to inform the workflow that this analytic activity is finished. When the execution is complete, Kafka is also informed that new results have been published and may be accessed by other analytic stacks. If there are dependencies on analytics B and C, they can be executed (note: these activities are not depicted).

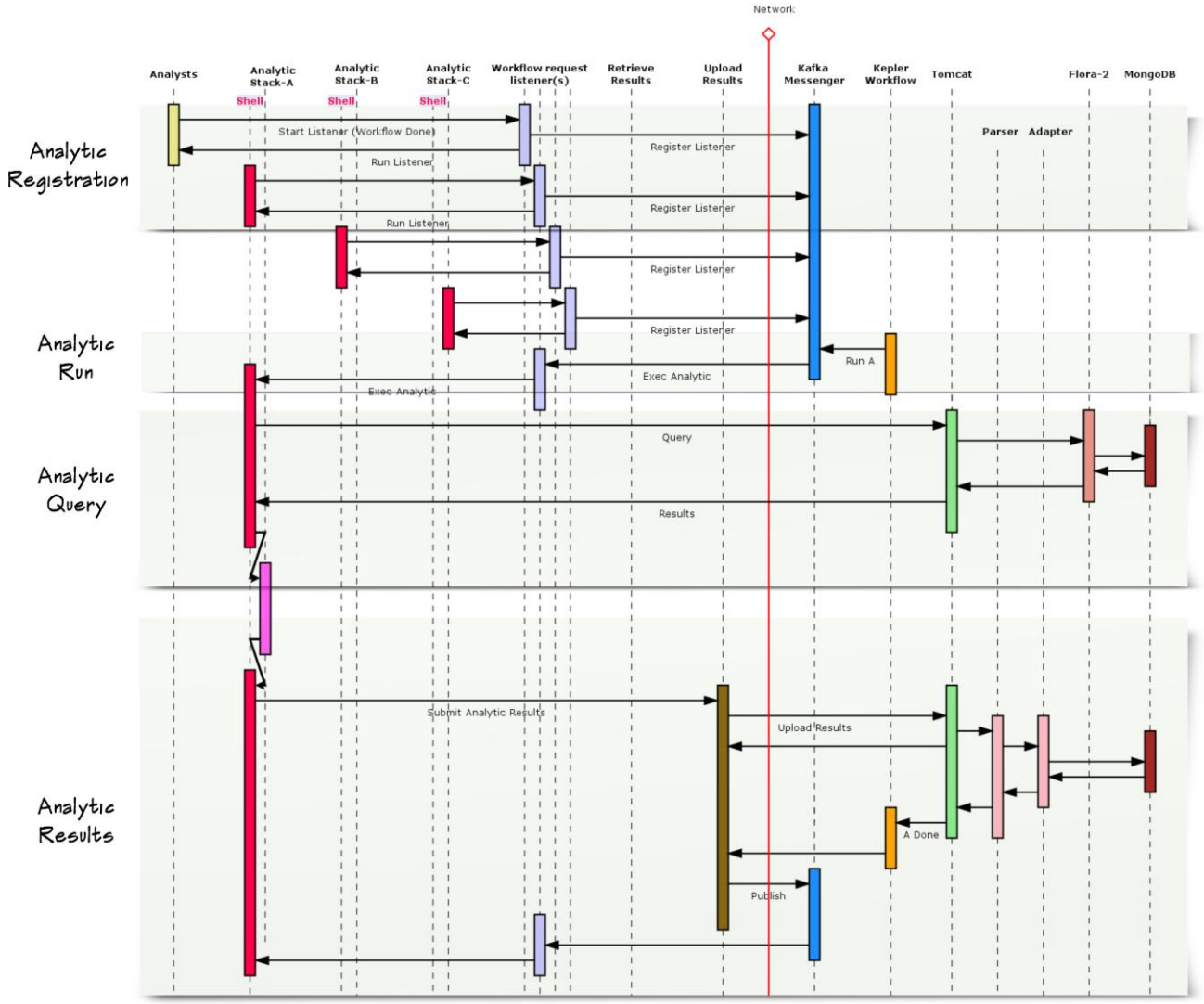

Figure 9: Analysis Exchange planned workflow

5. Case Study: Fraud, Waste, and Abuse

One of the initial applications for the Analysis Exchange is in support of a use case that seeks to uncover examples of fraud, waste, and abuse by performing data analysis drawing on sources and analytic capabilities from different companies. The specific example of fraud, waste, and abuse being explored focuses on fraudulent travel vouchers submitted to the U.S. Department of Veterans Affairs (VA), a topic that has incurred legal action in recent years against alleged defrauders [9]. Typically, veterans or claimants on behalf of veterans go to the Travel Benefits Office at a Veterans Affairs Medical Center and provide information about the distance traveled to get a voucher to submit for a reimbursement. Because this relies on honesty and accuracy of the travel expenses, it is easily targeted for abuse. This runs in the thousands of dollars per defendant, but it represents behaviors that are likely

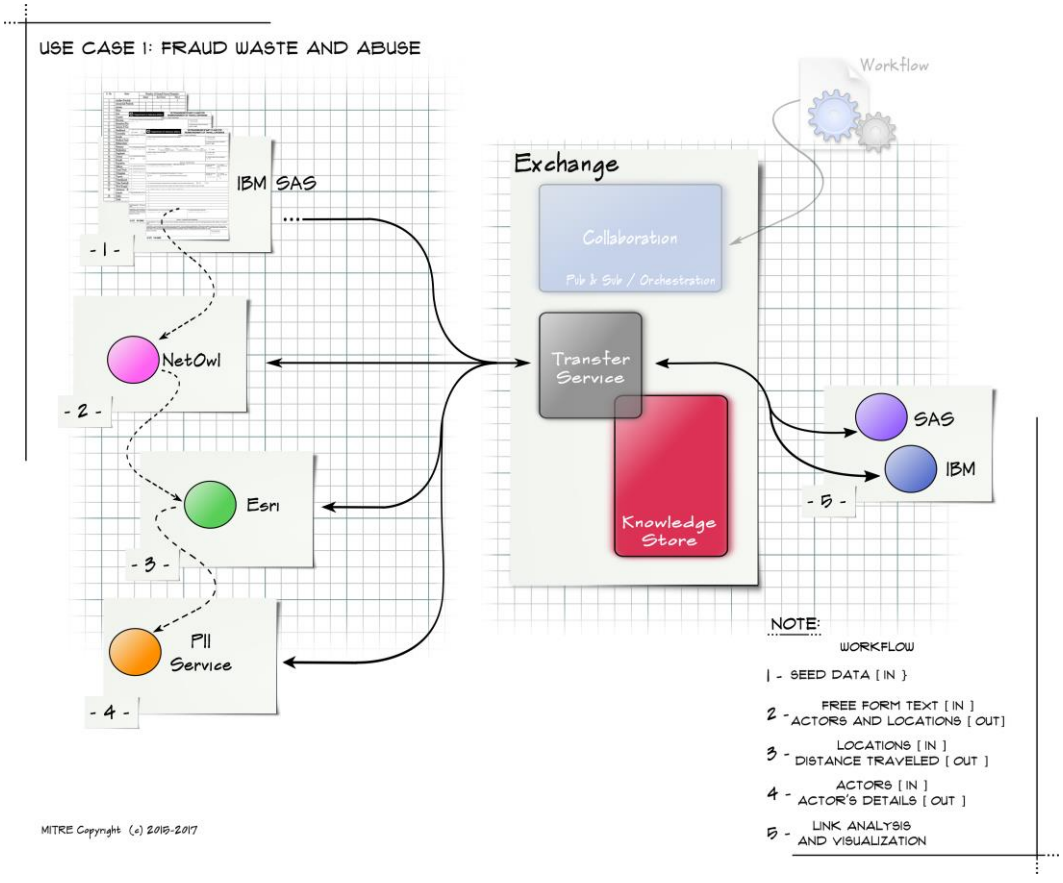

Figure 10: Analytic interaction with Analysis Exchange for the fraud, waste, and abuse use case

something that analysis of available data can detect. In this use case, we examine how a pipeline of analysis can achieve this, making use of the Analysis Exchange throughout.

The basic pipeline of analytic activity is expressed in Figure 10, and involves an industry partnership between IBM, SAS, NetOwl, and Esri. Seed data (1), which includes fields drawn from VA travel vouchers, VA records, and fixed information such as VA locations, is first ingest into the Analysis Exchange. These records have certain fields that may contain free text, and NetOwl (2) is invoked to break these fields down into constituent parts that better fit the expressiveness of the ontology. Each VA travel voucher represents one or two travel events between locations (typically a residence to a VA medical center and the reverse trip). Esri (3) is called to provide plausible paths between these locations to determine what is a reasonable mileage between the locations. Further enrichment (4) can also come from other sources that provide information on whether the veteran or claimant has an arrest or criminal record or has declared bankruptcy. All these factors together can undergo downstream statistical analysis or link analysis such as SAS or IBM provide (5). These activities can identify where there are significant differences from existing records or norms and flag cases for further review. Taken together, this pipeline results in an analysis product that can be rendered for a person to examine and make judgments either about individual records that are aberrant or a perspective at a higher level of problems across the records.

In each case described above, the flow the pipeline goes into and out of the Analysis Exchange. The paper has earlier discussed how transfers of information can be made through adaptation, and that pattern is applied here. Information from these sources can arrive in different formats (e.g., CSV records, XML). These inputs are parsed following their format and schema and then their contents are mapped into the Analysis Exchange Ontology, preserving the meaning behind the semantics as this information moves from the source data or analytic output to the common representation. This will involve creating GSON[5] objects based on the results that are then serialized into the JSON statements that are placed in the data store and encode the entities in the data and their relationships and properties. There is also need for query for information that must be drawn from the Analysis

[5] https://github.com/google/gson/blob/master/UserGuide.md

Exchange. For cases that require adaptation, the content will be offered in a form that is palatable to the analytics' expectations.

This work demonstrates the power of the interoperability of the Analysis Exchange. While five major components of the pipeline have been identified, this is quite extensible to new analytics or sources of enrichment. Components can even be completely replaced by alternatives if necessary or if new capabilities come online.

6. Glossary

Adapter – an Analysis Exchange component that converts results between a variety of native analytic models and a common data model with a supporting ontology.

Analytic stacks – software applications or services that analyze data and derive results and knowledge; in this context, these are expected to execute outside the Analysis Exchange.

Analysis Exchange – a network-enabled repository for results from external analytic services, fostering industry and government collaboration by facilitating knowledge fusion and downstream analysis.

Analysis Exchange Model 1.0 – the initial reference implementation of the Analysis Exchange architecture, as described in this document.

Analysis Exchange Ontology – the ontology capable of representing adapted analytic results from various sources and intended to support the use case generated for the prototype Analysis Exchange.

Analytic Technology Industry Roundtable – a group which brings together leaders of analysis and analytic technology companies to foster dialogue and achieve common goals (more information on the Roundtable can be found at www.mitre.org/roundtable).

API – application programming interface.

BFO – Basic Formal Ontology.

Collaboration Services – the Analysis Exchange components that drive and manage the workflow as analytic stacks interact with the Analysis Exchange.

Pub/Sub Service – the Analysis Exchange component that allows the publication of and subscription to analytic results in the Analysis Exchange; this is a principal element of the Collaboration Services.

Knowledge Store – the Analysis Exchange component that holds results produced by external analytic stacks.

Transfer Service – the Analysis Exchange component responsible for the transfer of results between the Analysis Exchange architecture and external software or users.

UUID – universally unique identifier.

VA – U.S. Department of Veterans Affairs.

Workflow – the sequence and flow of analytic stacks, which can be deployed and will execute on both source data and generated results.

Workflow instance – a specific execution of a workflow of analytic stacks on specific data.

7. References

[1] A. Etches, C. Brown and J. Stultz, "Analytics and Use Cases," 8 November 2016. [Online]. Available: http://www2.mitre.org/public/analytic-technology/pdfs/Analytics_and_Use_Cases_Study_IBM_SAS_11_25_16.pdf.

[2] B. Choung and M. Chandler, "Review and Study on Industry Collaboration," 8 November 2016. [Online]. Available: http://www2.mitre.org/public/analytic-technology/pdfs/Industry_Architecture_Survey_and_Review_Study_Palantir_11_25_16.pdf.

[3] R. Winder, J. Jubinski and A. O'Hanlon, "Analytic Architecture Survey and Review," 8 November 2016. [Online]. Available: http://www2.mitre.org/public/analytic-technology/pdfs/Architecture-Survey-Review.pdf.

[4] W. Niehaus, B. Adams, S. Panzer and L. Hanson, "Alignment of Capabilities," 8 November 2016. [Online]. Available: http://www2.mitre.org/public/analytic-technology/papers.html.

[5] R. Winder, N. Giles and J. Jubinski, "Implementation Recommendations for MOSAIC: A Workflow Architecture for Analytic Enrichment," McLean VA, 2011.

[6] R. Winder, J. Jubinski, J. Prange and N. Giles, "MOSAIC: a cohesive method for orchestrating discrete analytics in a distributed model," in *International Conference on Application of Natural Language to Information Systems*, Manchester, UK, 2013.

[7] R. Arp, B. Smith and A. Spear, Building Ontologies with Basic Formal Ontology, Cambridge, MA: The MIT Press, 2015.

[8] I. Niles and A. Pease, "Towards a standard upper ontology," in *Procedings of the International Conference on Formal Ontology in Information Systems*, 2001.

[9] A. Margulis, "Charges files against 16 veterans for hospital fraud," 1 July 2016. [Online]. Available: http://www.citizen-times.com/story/news/local/2016/07/01/charges-filed-against-16-vets-hospital-fraud/86591512/. [Accessed 27 June 2017].