

# Pedigree Management and Assessment in a Net-centric Environment

Marisa M. Gioioso\*, S. Daryl McCullough, Jennifer P. Cormier, Carla Marceau, Robert A. Joyce  
ATC-NY, 33 Thornwood Drive Suite 500, Ithaca NY, USA 14850

## ABSTRACT

Modern Defense strategy and execution is increasingly net-centric, making more information available more quickly. In this environment, the intelligence agent or warfighter must distinguish decision-quality information from potentially inaccurate, or even conflicting, pieces of information from multiple sources – often in time-critical situations. The Pedigree Management and Assessment Framework (PMAF) enables the publisher of information to record standard provenance metadata about the source, manner of collection, and the chain of modification of information as it passed through processing and/or assessment. In addition, the publisher can define and include other metadata relevant to quality assessment, such as domain-specific metadata about sensor accuracy or the organizational structure of agencies. PMAF stores this potentially enormous amount of metadata compactly and presents it to the user in an intuitive graphical format, together with PMAF-generated assessments that enable the user to quickly estimate information quality. PMAF has been created for a net-centric information management system; it can access pedigree information across communities of interest (COIs) and across network boundaries and will also be implemented in a Web Services environment.

**Keywords:** Information management, pedigree, provenance, net-centric, information quality

## 1. INTRODUCTION

Information pedigree, or provenance, is needed to establish trust in information, ensure accountability, discover sources of errors, and correct propagated errors. Many special-purpose pedigree management systems have been developed, yet a general-purpose, extensible system that can be used in a net-centric environment of disparate information management systems has remained an elusive goal. In this paper, we describe the Pedigree Management and Assessment Framework (PMAF), which implements such a system. PMAF supports distributed and extensible pedigrees, a sophisticated query framework, and automatic information quality assessment, as well as pedigree drill-down. PMAF was originally implemented for the Operational Information Management (OIM) [1] environment, an information management environment conceived and built by the Air Force Research Laboratory in Rome, NY, but it is portable to any environment, such as web services.

The Infosec Research Council in November 2005 listed Pedigree as one of eight important but hard problems in information security that researchers should focus on in the next five to ten years. Two of the challenges represented by pedigrees are the tremendous volume of pedigrees in a net-centric world and the need for fine granularity—that is, representing pedigrees of parts of composed information objects. PMAF addresses these through a novel representation of provenance, which is presented in Section 3.

To avoid an exponential explosion of pedigree data, PMAF supports distributed pedigrees through its *federated query* mechanism. Federated query and novel techniques to enhance query efficiency are described in Section 4.

Both people and applications refer to pedigree in order to assess information quality. In some cases, the assessment requires a drill-down into details of the pedigree. However, in many cases, PMAF is able to present quality information directly to the user or application. PMAF's automatic *information quality assessment* capability is described in Section 5. The user can use the pedigree browser interface, described in Section 6, to perform a deeper manual analysis on the pedigree.

Section 7 describes related work, and Section 8 presents conclusions and future plans.

\*mgioioso@atc-nycorp.com; phone 607 266-7105; <http://www.atc-nycorp.com>

## 2. PMAF CONCEPT OF OPERATIONS

Three challenges presented by pedigrees are: volume, granularity, and the need for domain-dependent metadata.

Volume is a major problem in an age when documents are created, modified, recomputed, transformed, or transferred numerous times. In each step, many details are added to the pedigree. Each time a document is transformed, if the pedigree metadata is copied from its prior version and then augmented by metadata detailing the most recent change, the pedigree grows quickly. Not only will the current document's pedigree be large, but it will overlap in all but the final step with the pedigree of its prior version. For example, a document's pedigree can specify that the document was composed or aggregated from two source documents, and those were each in turn composed from two source documents, and so forth. The pedigree can grow exponentially in this manner with respect to the number of transformations the document has gone through.

Another pedigree challenge is granularity of information. In a net-centric world, information in new information objects is often aggregated from multiple documents, images, databases, or other parts. But a pedigree of an information aggregate, while it may provide some value to the user, is in general not sufficient to assess quality and reliability. For example, if a document's pedigree lists three documents as sources, this is not sufficient information to determine whether the three documents are all sources for a single piece of information, in which case they corroborate one another, or for three separate pieces of information in the document, in which case they are independent of one another. Similarly, the three source documents carry different levels of significance if they are sources for a critical versus a marginal piece of information in the original document.

A third challenge in pedigree is the need for extensibility to accommodate domain-dependent metadata. It is not possible to make a list of all the "pedigree" information that anybody will ever want about any information object. Rather, the representation of pedigree must enable users to specify and query on new types of metadata about information objects.

The Pedigree Management and Assessment Framework (PMAF) addresses these three challenges by representing provenance metadata using a novel technology that supports metadata composition and extensibility.

### 2.1 PMAF architecture

PMAF is intended for situations involving the following participants:

1. One or more information object repositories. We assume that there is a unique naming system for objects obtained from each repository.
2. One or more repositories of provenance data for the information objects. We do not assume that the provenance data is stored in the same repository as object information.
3. A client application that reads and writes information objects and the corresponding provenance data.
4. PMAF, which provides an API for both querying and publishing provenance data.

Our design for PMAF and how it interacts with our participants is shown in Figure 1 and the components described below.

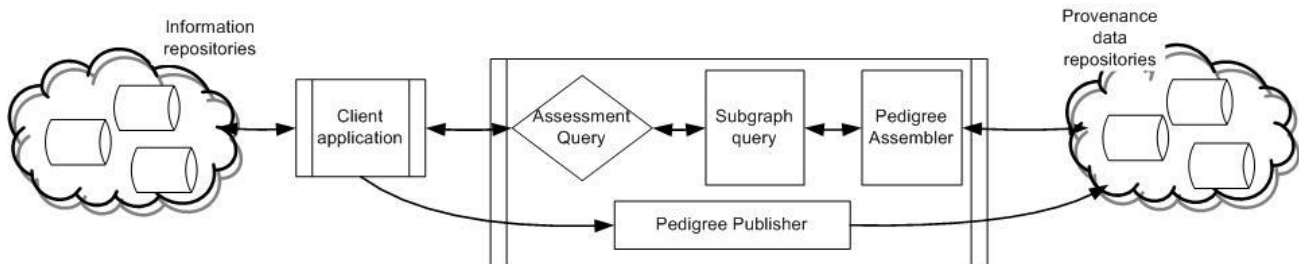


Figure 1: PMAF Architecture

**Assessment query** - A typical PMAF Client receives objects from the object repository and invokes *Assessment Queries* to help determine the quality and trustworthiness of this information.

**Pedigree publisher** - The Client may also publish objects to the object repositories, and make calls on PMAF to publish corresponding additions to the provenance data.

**Subgraph query** - Each assessment invokes one or more *Subgraph Queries* for the given object. The result of a subgraph query is to produce a provenance subgraph, a data structure describing a particular aspect of the provenance of an object. Each request for a subgraph includes a specification for the type of subgraph desired, together with a URI that uniquely names the subject of the subgraph. The Subgraph Query component constructs the requested subgraph out of root pedigrees it receives from the Pedigree Assembler.

**Pedigree Assembler** - The *Pedigree Assembler* assembles the root pedigrees from pedigree fragments that it receives from one or more repositories for provenance data. Our design for the pedigree assembler supports extensibility through the use of pluggable modules to handle system-specific operations that depend on the type of information repositories being used, and to use a standard framework for encoding provenance information, regardless of the type of provenance storage being used. The Pedigree Assembler can easily be adapted for different types of provenance storage, and of different granularities of provenance data. Provenance data could be stored in files, or in a database, or as separate objects in an information object repository. Multiple sources of provenance data must be merged before full provenance can be assembled. The Subgraph Query component does not depend on any of these details handled by the Pedigree Assembler.

**Publish API** - The mapping between information object names and the corresponding provenance data is a modular component that can be plugged into the PMAF framework to accommodate new types of information object repositories. The *Publish API* allows client applications to publish provenance data for new objects without needing to know the format of the provenance data or where it will be stored. It makes use of the modular mapping between object names and provenance data.

### 3. PROVENANCE MODEL

The PMAF model of pedigrees includes the following concepts.

**Resource** - A *resource* is any nameable object that can have an associated pedigree, or can be referred to in a pedigree. People, web sites, documents, programs, etc. are all considered resources.

**Provenance data** is information about how a resource was created, transformed, or used.

**Pedigree fragment** - We assume that provenance data is stored in one or more provenance repositories in the form of *pedigree fragments*. Each pedigree fragment contains information about a resource. A pedigree fragment consists of one or more “local” claims about the provenance of a resource. A local claim is one that involves only a single step of the creation, transformation, or use of a resource. For example, the facts that a document was created (1) by a particular person, (2) at a particular location, (3) by running a particular program, and (4) using particular source documents are all local facts about that document. In contrast, if document A was created using document B as a source, then the sources of document B will be considered *nonlocal* facts about A. The complete pedigree of a document is assembled from local statements about that document, its sources, the sources of those sources, etc. Typically, the pedigree fragments for a resource will be published by the same client application that created or modified the resource. However, there are also possible “third party” contributions to the pedigree. These are pedigree fragments created by an application that is neither the writer nor the reader of the associated resource document. Allowing such third-party contributions is important in dealing with objects whose sources are unknown (such as articles found on the Web). Whether the pedigree fragments are published by the creating application, or by a third-party application, provenance data will typically be published through the PMAF Publishing API.

**Root pedigree** - A provenance repository can be queried to get pedigree fragments associated with a given resource. These pedigree fragments are assembled into a *root pedigree* for a resource. A root pedigree is a complete collection (complete relative to the information available, of course) of local information associated with a resource. We consider the complete root pedigree for a resource to be divided into two parts: source record and usage record. The source record for a resource is the local history of a creation or transformation step resulting in the current state of the resource. The usage record for a resource describes how that resource is used in the source pedigrees of other resources. For example, if document A is produced using document B as a source, then this fact will be in the source record of A, but will be in the usage record of B. Typically, the usage records can be more difficult to assemble than the source records, because the

source record only needs to be created once, when the resource is created, while the usage record can involve many, many different individual creation or transformation steps performed by different users at different locations and times.

**Provenance subgraph** - A *provenance subgraph* is a collection of related provenance statements assembled for use by some assessment procedure. Like a root pedigree, a provenance subgraph is associated with a particular resource, but it may include information from several connected root pedigrees. For instance, a *source pedigree subgraph* for a resource A may include statements of the form “A has source B” as well as “B has source C”. A provenance subgraph thus allows one to draw nonlocal conclusions about a resource, such as “foreign news reports contributed to this document”. There are two pure approaches to creating such provenance subgraphs: (1) Create a complete pedigree, and then filter to get the desired provenance subgraph. (2) Build up the desired provenance subgraph by gathering pedigree fragments as needed. Because complete pedigrees can be huge and can be distributed among many different repositories, we consider it impractical to ever create a “complete” pedigree, so PMAF follows the latter approach. We use the word “subgraph” because conceptually these objects can be thought of as subsets of the complete pedigree, which in turn can be thought of as a potentially huge graph of relationships among resources. In PMAF we describe these relationships using RDF (discussed below).

PMAF uses a two-stage query system to assemble provenance information for use by assessments. The first stage involves querying the provenance repositories for pedigree fragments associated with a given resource. The results of this stage of query are assembled into root pedigrees. The second stage of query involves assembling appropriate root pedigrees together to make a required provenance subgraph.

### 3.1 Representing provenance using RDF

In PMAF, pedigree fragments, root pedigrees and provenance subgraphs are all represented as graphs, described using the Resource Description Framework (RDF) specification [2]. RDF supports the unique identification of entities, the exchange of information at a machine-readable level, the unlimited extensibility of terminology used to represent pedigree, and the semantic tagging and subsequent inference and other processing of the pedigree information. RDF represents information as a collection of statements, called triples. Each triple has three elements – the subject, predicate and object. Each element in the triple is specified by either a Universal Resource Identifier (URI) or a literal string. A collection of RDF triples can be viewed as a graph, in which the subject and object are nodes and the predicate is a directed arc that points from the subject to the object.

RDF is a standard framework for encoding metadata similar to XML. Like XML, RDF is a format for representing information about resources that reside on the web or are identified on the web by URIs. URIs can be used to uniquely describe anything – any information that exists on the World Wide Web (in the form of URIs) as well as any concepts, persons, or objects that do not exist in the WWW. So if related pieces of information exist in separate locations on a server or on the WWW, the relationships among them can be detailed using RDF or XML, and the URI, like a reference in a paper, can be listed instead of the full set of information the URI refers to. For example, a sensor may generate thousands of images an hour. The make and age of the sensor may help a user determine the reliability of the image that was taken, but each image does not need to carry around a copy of the make and age metadata of the sensor in its pedigree metadata. Instead, that sensor metadata can be placed in one unique location and only that location’s URI will be enclosed in each image’s pedigree metadata. In addition, once an image has been transformed or fused with other information, the resulting image need not carry around another copy of the URI referring to that sensor metadata; instead, it will carry a URI to the previous image’s metadata pedigree portion, which in turn carries the sensor metadata URI. In this way, the volume has been reduced to a linear function of the number of transformations.

Once someone requests and is granted access to information from another domain, he can create new documents based on the queried one and extend the pedigree in his own domain. In this case, part of the pedigree resides in the original domain, and part (the part relating to the new modifications) resides in the new domain. RDF is ideally suited for such cross-domain pedigree storage for two reasons.

First, whereas XML represents a tree structure with a root node and nested layers, RDF represents a graph, with any piece of information connecting from and connecting to multiple elements. Because of this, we can record pedigree and descendancy; we can represent an information object as being derived or composed from multiple objects as well as being used as a reference for multiple objects.

Second, RDF allows ontological reasoning on identifiers. For an inter-domain information exchange, this utility can be used to translate between the terminology and standards of the different domains. Using inference allows for the

possibility of storing large amounts of provenance data in a very compact form. For example, if a sensor produces thousands of reports, once every 10 minutes, it is not necessary to create pedigrees for each report. Instead RDF would allow a single statement of the form “All reports of type T were created by sensor S”. This information would allow a pedigree for a particular report to be generated on the fly, as needed.

#### 4. FEDERATED QUERY AND PUBLISHING

Two-stage queries, as described above, first generate candidate pedigree fragments by searching the system-specific provenance data of the host system, then leverage that subset of pedigree to run a more focused and efficient second-stage RDF query. Without this staged pairing down of search candidates, each single search would be a rote and inefficient task. For a detailed discussion of subgraph querying, see [9].

PMAF was designed with a pluggable interface to reside on any kind of server or service, and to query RDF pedigree in any kind of data management system. In addition, for any given domain that includes various servers and data management systems, PMAF can uniquely identify any piece of pedigree. This design is made possible through the definition of two system-specific functions - the *server connection* class, which defines how to access pedigree in each repository, and the *information-pedigree association* class, which associates the URI of an object with the URI(s) of its corresponding pedigree(s). Once these definition classes are provided, PMAF can make a federated query over a set of disparate systems.

Locating pedigree elements across multiple systems of possibly different connection and storage formats is made possible through the use of URIs. Using the URI specification guidelines [7] available through the W3C, we developed a URI scheme format to identify and retrieve pedigree data on our sample storage environments – the AFRL OIM server, and an RDF triplestore database. For example, the URI format for a pedigree stored in an OIM server is:

```
oim://<server name>/<object type>/<object version>/?<metadata query string>#<pedigree object fragment identifier>
```

The Java URI handler functions enable this format to be recognized and processed, providing definitions for how to connect to the OIM server, how to translate and submit the query string to the server for processing, and how to interpret the return data. The Jena [8] RDF query engine at the core of the PMAF automatically searches out these definition classes to retrieve and interpret any specified data type. This feature allows PMAF to make federated queries seamlessly. Additionally, pedigree source references (or any other kinds of pedigree links) can uniquely reference pedigree in other systems, and PMAF can follow those links from one system to another. In this way, the user can make a deep pedigree search that goes from an active server to an archive server for older sources.

For example, a user may trust only information from images that are over a certain threshold resolution and that came from sensors of a certain model and year. The pedigree identifier of the raw image may be on one server in a file system, and it can uniquely refer to the sensor features that are available on a separate host in a database. The query can follow the links from one place to another to satisfy the query and then return any combination of information from both locations and formats into an RDF subgraph for local viewing or analysis.

In addition to host and data storage definitions, PMAF will look for definition classes that specify the association between information objects, such as documents or records, and their corresponding pedigrees. This happens during a typical query because in general, PMAF can accept query URIs that identify an information object rather than its pedigree. Thus the calling application does not need to know the URI of the pedigree, just of the object it describes. Then the information-pedigree association class is called to construct the pedigree URI(s) identifying the pedigree associated with the requested object. For a given system, a system administrator may define, for example, that information of a certain type (specified in the object’s URI) is stored behind a web service, and its pedigree is stored in a database on some server A. The query passed to PMAF would specify the URI of the information – including web service protocol, information object type, and other identifiers. The association class would use these identifiers to associate this information with pedigree from the database, and construct the appropriate URI for it. That pedigree URI would then be passed on to the server connection class to connect to the database and submit the query.

##### 4.1 Querying using RDF

PMAF includes a core provenance ontology based on and extended from the Dublin Core Metadata initiative [14]. Using the RDF-S class hierarchy, we refined existing Dublin Core terms to ones related to a decision-making process, the

dependency of data in general, and to the occurrence of conflicting data, all relevant to Air Tasking Orders and sensor fusion.

By using RDF, PMAF can take advantage of existing RDF query language systems such as SPARQL [9]. When all provenance information is centralized in one RDF provenance repository, SPARQL queries are efficient and sufficient to make expressive queries. SPARQL cannot be used, however, to search for information that resides on multiple provenance repositories. For this reason, we need to use of a separate Pedigree Assembler (as described above) that gathers together pedigree fragments from multiple provenance repositories before SPARQL can be used.

## 4.2 Publishing and storing provenance data

The Pedigree Publisher provides pedigree writer classes that create pedigree data in the form of a collection of RDF triples. Although there is nothing in the PMAF API that explicitly enforces the following standard, we assume that a pedigree fragment for an information object consists of statements with the pertinent information object as the subject, and information that came before it as the object. It is a reasonable model to assume because during the creation of a new information object A, the pedigree of A should be created along with it, and would refer back to all the information {B1, B2,...Bn} used to create it. Referring to information provenance is a process that looks backwards in time, so in this way, most pedigree statements are backwards-looking.

PMAF does not specify the pedigree storage mechanism. A system administrator may append the pedigree to the metadata of the information object, or specify alternate locations for the pedigree. PMAF only assumes that for each piece of information, there is at least one pedigree root node, which names that piece of information's URI. As long as that node's URI is well-defined, PMAF can query for and generate pedigree for that information object.

If an information object has editable and extensible metadata, then it can be augmented to include the source record for the information that the object contains. The source record consists of RDF triples describing the sources used to update the information object. These source objects are described by giving their URIs. To compute the extended pedigree it is only necessary to look up those source objects and their associated source records. If the information objects are read-only, then separate objects can hold the source records. If information objects are numerous and always from the same information source—such as messages passed in a peer-to-peer system that each carry a sensor reading—then a pedigree can be generated for the class of objects instead of each instance of that object. In this case, a static location can be defined to house the pedigree associated with information objects of that type.

The URIs in the pedigree are well-defined as long as they refer to a single information object. There are several formats for which it is trivial to name an identifier – such as a record in a database that has a unique key, a java object in memory that has an instance identifier, a file that has a unique file name, or a part of an XML document that has a unique XML tag for that part. Other cases remain as research problems, such as uniquely identifying a part of an unstructured text document or a response from a web service with no persistent state.

In particular, the unstructured text case is difficult because of the requirement to use fine-grained pointers. An unstructured document can be divided up and labeled by its syntactic parts – sentences, phrases, words – but this method would not identify the semantic parts. For now, PMAF only supports XML tag fragment identifiers.

It is more difficult to associate a pedigree for information that is non-persistent, such as typical web service messages, which are generated on the fly in response to requests. A client can make a request to the web service for some information and may want to also retrieve the pedigree of that information. At the time of response, it may be possible to identify the sources or processes that produced the return result, but it would not be possible to query the web service for this pedigree after the fact. For now, PMAF will support only web services for which the return information would persist on the local client. In that case, the pedigree would start on local and could point back to the background information at the back-end of the web service.

## 4.3 Performance enhancements

PMAF does not attempt to store a complete pedigree for each information object, but instead assembles the relevant pedigree information on the fly. While this approach greatly cuts on the volume needed to store pedigrees, it correspondingly increases the time required to process pedigree queries. For this reason, we have taken measures to improve query response times such as caching or pre-fetching pedigree when its possible use is anticipated.

Once a pedigree fragment has been retrieved from a repository, a copy is saved in the local cache, and when a new query is invoked this local cache is searched first before searching remote repositories. A drawback of using a cache is that it

can quickly become out-of-date. For example, if a document A has a usage record recorded in the cache, and later, a new document is created that uses A as a source, then A's cache representation is no longer complete. To prevent caches from getting too out-of-date, PMAF periodically, or upon client application trigger, clears or updates the cache.

Even before the user requests pedigree information, the client application can use the PMAF API to pre-fetch the pedigree fragments and store them in local cache. The client application can determine what triggers pre-fetching of pedigree fragments, such as a user's request for information in a case where pedigree would typically be requested soon after.

## 5. ASSESSMENT

In time-critical situations, warfighters and others need to quickly evaluate information quality; there is no time to drill down through a large pedigree. To support quality assessment, PMAF offers quick, accurate and pointed analyses based on the pedigree. Documents and relationships in the pedigree that have an immediate bearing on the quality of the information are evaluated and presented to the user. Other analyses estimate the trustworthiness of information publishers or the reliability of documents.

Several groups have posed and tested various approaches to metrics for quality, focusing on trust in particular. Example approaches include the eBay metric [11], the TidalTrust algorithm [12], the Appleseed metric [13] and PageRank used in the Google search engine. In each of these cases, a score is calculated to quantify the relevance of a piece of information or the trust in a publisher of information.

A score or batting average seems desirable because it is simple and straightforward. However, such a simple metric is not possible in the multi-dimensional world of information quality. Instead, PMAF provides the user with five metrics for a document or information object that help him to evaluate its quality: its *influence* on other documents; whether it has been superseded or *deprecated*; whether it derives from *conflicting sources*; publisher *reputation*; and *corroboration analysis*. Based on the assessment analyses, the PMAF user can estimate the quality of the information.

### 5.1 Impact

The "impact" assessment describes the influence a document has on its domain. Frequent use often implies high quality. Documents that claim a given document A as their source serve as implicit votes of confidence in A. Human users who publish reports based on an intelligence document B clearly feel that B contains valuable intelligence. In other cases, frequency of use indicates relative importance. For example, if intelligent agents generate fifty alerts based on the content of sensor feed C, but only two alerts based on sensor feed D, feed C may be placed at a battle location, while feed D may include relatively empty frames with little activity. Note that the impact assessment is based on an analysis, not of provenance, but of the *descendants* of the information – that is, information created later that claims a given document as a source.

### 5.2 Deprecation

The "deprecation" or "age" assessment qualifies a document as either being up-to-date or deprecated – either out-of-date or not accurate. For example, updated sensor reports from a certain area invalidate older sensor reports, and the new release of a report may correct an error in a previous version. In either case, documents that rely on the older information may be invalidated. In a scenario on mission replanning that we investigated, a planned mission was invalidated by new sensor reports revealing a different object than previously identified. In such cases, it is crucial for mission planners to discover the error as soon as possible. PMAF identifies outdated documents by looking for replacement documents.

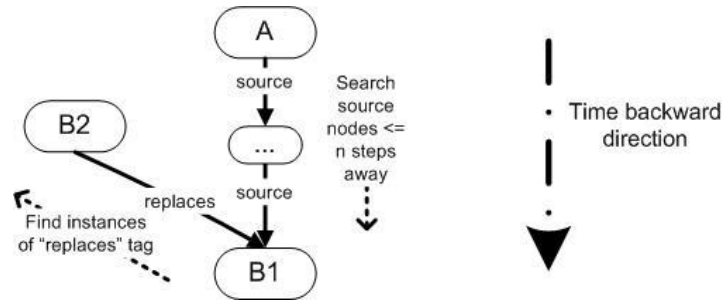


Figure 2: Deprecation query

### 5.3 Conflict notification

The “conflict notification” assessment notifies the user that an information object comes from sources which were found to be in conflict by some process external to PMAF. To handle conflicts, PMAF requires the pedigree publisher to provide a conflict assertion, the set of information objects in conflict, and the resolution (e.g. that one information object was preferred or that a new object was created combining aspects of the conflicting objects). PMAF locates these items in the pedigree, if they exist and notifies the user of their existence and pertinent details.

### 5.4 User feedback

Reputation systems based on consumer feedback are familiar to everyone from Internet examples, such as Amazon book rankings and evaluations and eBay seller reputations. PMAF enables its users to evaluate the quality of documents and records the evaluations with the information pedigree. The precise form of the evaluation is domain-dependent.

The PMAF sample feedback form addresses some problems often associated with feedback systems: unstructured feedback, bias, and feedback granularity. To support explicit feedback, a preferred environment would allow different forms for different classes of users. Experts would be allowed a looser format and the ability to comment on an information object at a coarser level of granularity. A lower level user might be allowed to provide feedback only on those information objects or assumptions whose correctness he can judge. In that vein, the sample feedback form includes fields to specify at a fine-grained level the information the user is providing feedback for – for example, an XPATH expression in an XML document. His rating applies only to that item. (It may be aggregated into the overall rating of the information object, but that aggregation is not trivial, and probably depends on the type of information.) In the future, PMAF will support automatic domain-specific sub-document identification; for example, the user will be able to click on an item in a document map and PMAF will automatically identify the intended part of the document. (In non-XML documents, identifying document parts is not always feasible using current technology.)

A unique identifier for each user in the pedigree system enables PMAF to identify the user who provided any given feedback item and retrieve other information about him.

### 5.5 Corroboration analysis

Corroboration analysis provides deeper insight into apparently corroborating sources. Information is given more credibility when it originates from several corroborating sources. But in many cases, multiple corroborating sources can themselves be derived from the same ultimate source. For example, the Associated Press disseminates stories that are incorporated into several newspapers – The New York Times, the Boston Globe, etc. Upon seeing the same story in multiple newspapers, a reader may place more trust in it. But the reader would be deceived by not knowing that all stories came from only one ultimate source, the AP. In this case, the assessment of quality should not be based on the number of sources, but only on the credibility of the ultimate source, the AP.

When run, this assessment determines the “ultimate sources” of a given document – those for which no further source is available. The user can then inspect the original sources.

## 6. BROWSER INTERFACE

PMAF includes several graphical interfaces. Some support pedigree browsing and drill-down. Others enable users to quickly estimate information quality based on the assessments described in Section 5.

PMAF's built-in pedigree browsers provide three distinct views of the pedigree. The tree browser (see Figure 4) depicts the ancestry of a given document (or, alternatively, its descendants) in hierarchical form, where each node in the tree represents a source (or descendant) document. Such hierarchal trees are familiar to computer users from file system browsers. The graph browser (see Figure 3) depicts the pedigree as a graph, with labeled arcs to source documents, publishers, and other pedigree metadata. PMAF graphs are constructed using Isaviz [15], an RDF graphical tool. The graph viewer enables the user to zoom out to a bird's eye view of the whole graph or zoom in (drill down) to sections of interest. In addition to the built-in browsers, PMAF supports hooks for domain-specific visualizations of pedigree metadata. This is done by extracting data from the pedigree and inserting it into an HTML document. Figure 2 shows an example from pedigree data associated with a mission object in an Air Tasking Order.



Figure 3: Graph view and domain-specific mission view

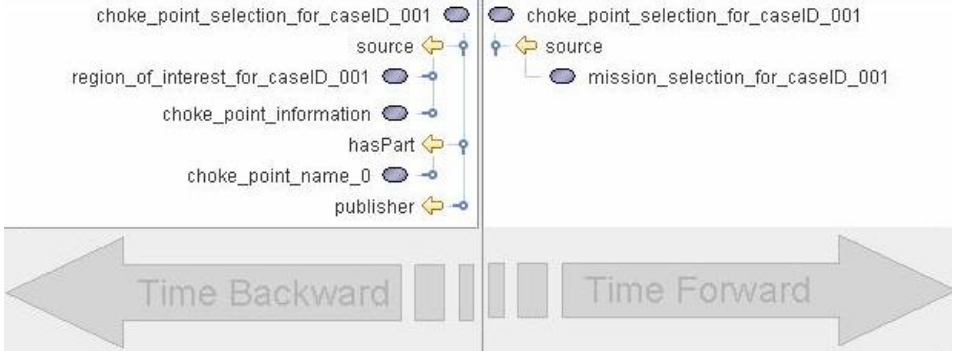


Figure 4: Tree view

### 7. RELATED WORK

Stuckenschmidt et al. [3] address the problem of evaluating RDF queries over a distributed graph and show how to optimize evaluation when the storage devices are all relational databases. When uniform data storage methods can be assumed, multiple join optimization techniques can be employed to optimize queries. Unfortunately, this is often not the case.

Several researchers [4, 5, 6] have represented pedigrees using RDF. However, their results are of limited use to us, since they assume the information, as well as the pedigree metadata, is encoded in RDF.

### 8. CONCLUSIONS AND FUTURE WORK

PMAF is a work in progress in both the fields of information provenance and the representation and operation of information using RDF. Our purpose is to create a system that describes information in any format, but we use RDF to represent provenance information. An important goal throughout this was to create an API that makes it simple to write

new provenance data and to take advantage of the query and inference capabilities of RDF, but all with an interface for future system administrators to take advantage of without complete knowledge or understanding of RDF. The next step here is to develop new assessments that are able to use numerical data to calculate statistical confidence estimates on information quality.

The current effort made portability and a distributed query a focus. Various distributed hosts with databases, file system storage and for server-client or service-type interactions were included in this portability model. The next step is to investigate what assumptions we must add or subtract to port to a peer-to-peer environment keeping the same goal of usability.

For our future research, we will incorporate inferencing into the query and assessment processes. As a simple example, we may want to query for all objects that are related to some node by a specified class of relationships. The class of relationships need not be explicitly defined in the query, but can be inferred based on an ontology of terms classifying such relationships. In another example, we may further enable querying across protocols, and consequently across communities of interest, by allowing federated searches that translate the terminology of the query into the terminologies of the candidate domains for matches.

## 9. ACKNOWLEDGEMENTS

Support for the research described herein has been provided by Air Force Research Laboratory contract FA8750-06-C-0023. Thanks to Dr. Paul Thompson of Dartmouth College for his analysis of statistical metrics.

## REFERENCES

1. Air Force Research Laboratory Information Directorate (AFRL/IF) Operational Information Management program (formerly the Joint Battlespace Infosphere), <http://www.rl.af.mil/programs/jbi/>
2. "RDF Primer", <http://w3c.org/2001/>.
3. H. Stuckenschmidt, R. Vdovjak, G. Houben, J. Broekstra, "Index Structures and Algorithms for Querying Distributed RDF Repositories", Proceedings of the 13<sup>th</sup> international conference on World Wide Web, 2004.
4. J. J. Carroll, C. Bizer, P. Hayes, P. Stickler, "Named graphs, provenance and trust", Proceedings of the 14th international conference on World Wide Web, May, 2005.
5. L. Ding, T. Finin, Y. Peng, P. Pinheiro da Silva, D. McGuinness, "Tracking RDF Graph Provenance using RDF Molecules", [http://ebiquity.umbc.edu/file\\_directory/papers/178.pdf](http://ebiquity.umbc.edu/file_directory/papers/178.pdf).
6. L. Ding, P. Kolari, T. Finin, A. Joshi, Y. Peng, Y. Yesha, "On Homeland Security and the Semantic Web: A Provenance and Trust Aware Inference Framework", <http://ebiquity.umbc.edu/get/a/publication/154.pdf>.
7. T. Berners-Lee, "Uniform Resource Identifiers (URI): Generic Syntax," <http://www.ietf.org/rfc/rfc2396.txt>.
8. Jena Semantic Web Framework, <http://jena.sourceforge.net/>.
9. "SPARQL Query Language for RDF", <http://www.w3.org/TR/rdf-sparql-query/>.
10. A. Barnell, "RDF Objects," <http://www.hpl.hp.com/techreports/2002/HPL-2002-315.pdf>.
11. A. Josang, R. Ismail, C. Boyd. "A Survey of Trust and Reputation Systems for Online Service Provision" *Decision Support Systems* 2006.
12. J. Golbeck, *Computing and Applying Trust in Web-Based Social Networks*, PhD Thesis, University of Maryland, 2005.
13. C. Ziegler and G. Lausen, "Spreading Activation Models for Trust Propagation," in Proceedings of IEEE International Conference on e-Technology, e-Commerce and e-Service (EE'04), 2004.
14. "Dublin Core Metadata Initiative", <http://dublincore.org/>.
15. Emmanuel Pietriga, "Isaviz: A Visual Authoring Tool for RDF", <http://www.w3.org/2001/11/IsaViz/>.