Preprint
UCRL- TR-202217

Design of the National Bioforensics Library Infrastructure

Tom Slezak, Terence Critchlow, Susan Hazlett, Nagiza Samatova, Praveen Chandramohan, Ramya Krishnamurthy

January 2004

U.S. Department of Energy

Lawrence
Livermore
National
Laboratory

DISCLAIMER

This work was performed under the auspices of the U.S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

Design of the National Bioforensics Library Infrastructure

Tom Slezak
Terence Critchlow
Susan Hazlett

Lawrence Livermore National Laboratory

Nagiza Samatova
Praveen Chandramohan
Ramya Krishnamurthy

Oak Ridge National Laboratory

January, 2004

Introduction

This design document is the first concrete step in developing a national resource for Bioforensics, the *Bioforensics Information Encyclopedia.* This resource will contain a semantically consistent representation of all the information relevant to the nation's bio-defense efforts. The availability of such a resource will provide analysts and scientists with efficient, timely access to the information needed to accomplish such tasks as locating relevant experts, determining an optimal order of appropriate forensic tests, and comparing the "signature" of the current attack to previous ones. Much of the information required for this repository is scattered across multiple institutions in differing formats and is either currently being generated or is proposed as part of new research efforts. Without effective integration and curation, the data is of limited availability and use. This resource would be utilized by several government agencies to prepare for, investigate, analyze, and respond to bio-terrorist events. In particular, NBACC requires a highly advanced knowledge base of this nature to perform its unique role for the Department of Homeland Security (DHS).

Under our current execution plan, an implementation of the design provided in this document will be demonstrated in a limited functionality prototype that can be deployed at NBACC-West (LLNL). This prototype will provide analysts with access to semantically consistent data from multiple data sources through a single, web-based interface. This interface will support a limited number of queries against an underlying data warehouse. The prototype data warehouse will include information from at least three of the scientific domains relevant to Bioforensics (e.g. genomics, pathogen signature, disease symptoms, chemical / toxicity). While the purpose of this document is to provide a design for the entire infrastructure, we identify those aspects of the infrastructure that will not be included in the initial prototype in Section 0.

This document is considered to be a living document. As a result, we expect to incorporate changes in the infrastructure design as we learn from our initial prototypes, explore additional collaborative opportunities, and interact with our target customers. We also expect to extend this document with the inclusion of appendices focusing on the detailed design and implementation strategies in the coming year.

System Overview

Our approach, as shown in Figure 1, extends the traditional data-warehousing model to support dynamic, complex, multi-modal information accessed through a variety of user interfaces. Data warehousing is a common approach to presenting a unified view of disparate data from multiple data sources. In a data warehouse, all of the relevant information contained in a data source is translated to a consistent representation and explicitly stored on a local machine. While a warehouse may not contain the most recent information (e.g. if information has been added to a source since the warehouse has been refreshed), by storing the data locally, it provides a reliable and secure infrastructure. Given the relatively low update rates for the majority of information expected to be in this repository, this is not seen as a significant issue. Transferring the data from the source representation to the warehouse representation requires identifying the data that needs to be transferred (e.g. newly added or modified data), reading the data from the data source, performing the necessary transformations on the data to convert it to the desired representation, and entering it

into the warehouse repository. Performing these tasks requires computer scientists working closely with domain experts to ensure the resulting implementation reads and translates the data correctly.

Once the data is contained within a data warehouse, current technology allows analysts to perform queries by either directly querying the data warehouse or by using customized user interfaces to execute well-defined queries. Due to the amount of information, differences in granularity, and complexity of relationships contained within the warehouse, performing queries directly against the warehouse requires a more detailed understanding of the data than typical analysts would have. Customized interfaces provide an intuitive way for analysts to ask queries, but traditionally are not easily extended to ask new or complex queries and thus may lack desired capabilities.

To address this issue, our design places an abstraction layer between the data warehouse and the user interface. This layer defines a set of high-level abstractions that hide some of the underlying data complexity while maintaining the relationships between the concepts. In the near term, custom query interfaces that use these abstractions to obtain information from the data warehouse will be used. In the longer term, flexible interfaces will be created to allow analysts to interact directly with these abstractions through an intuitive graphical interface and form complex queries across these abstractions. At the present time, it appears that this high-level abstraction layer will be the semantic network system developed by Bob Burleson at LLNL.

While Section 0, System Architecture, provides additional details about the components that are used to implement this architecture, the remainder of this section provides a functional overview of (1) the data integration function, specifically, what is involved in populating the warehouse, (2) the types of queries that this architecture can support, and (3) supporting the data exploration capabilities, specifically, what needs to be done to execute a query.

Data Integration Overview

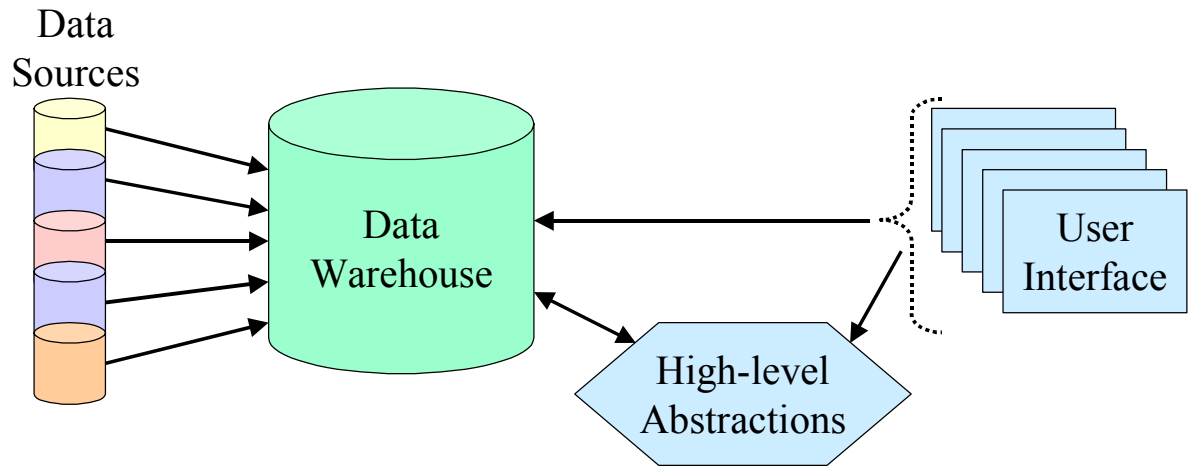

Figure 1 High-level overview of architecture

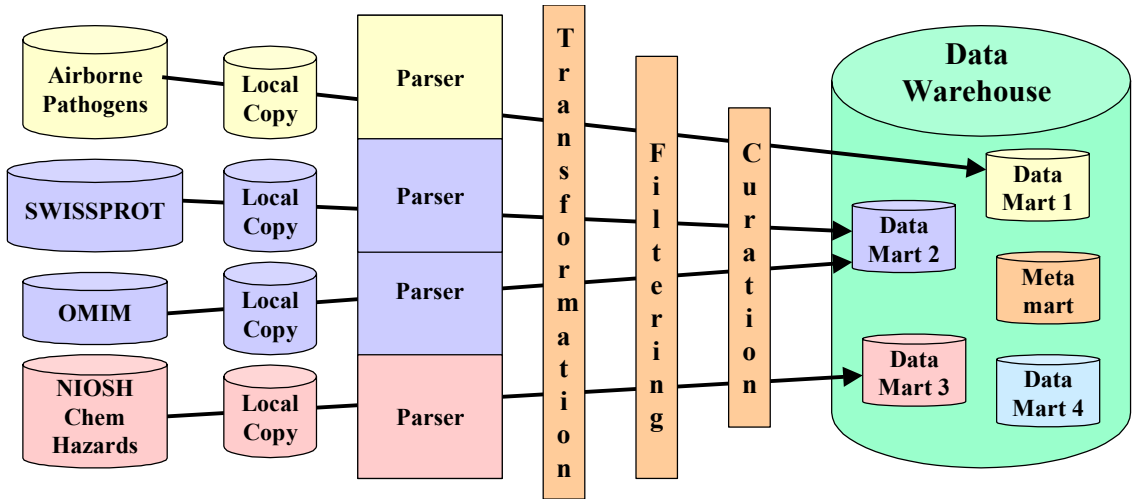

Figure 2 Data Integration Capabilities

As shown in Figure 2, the data integration phase requires more than simply identifying the data sources to be incorporated into the data warehouse. The data must first be identified and then replicated locally. Next, the data is parsed using a source specific parser and transformed from its original format into the globally consistent format used by the data warehouse. Because not all of the information contained within a data source may be relevant to bioforensics, some of the data may be filtered out either automatically or by a curator. The data curator may also update the data by either modifying it directly or annotating it with additional information. Due to the importance of keeping track of the lineage of all of the information in the warehouse, including any modifications performed by the curator, the warehouse will include appropriate data provenance information and tracking capabilities.

Given the number of distinct but related application domains that need to be incorporated into the warehouse, and the incredible complexity of each domain, the warehouse is implemented as a collection of data-marts instead of a monolithic entity. To ensure the appropriate relationships between the domains are maintained, a special, meta-data data-mart will be used to track the appropriate set of meta-data. This approach has several advantages, including:

(1) Each data-mart represents an integrated view of a single application domain. This makes it simpler to define a view of the data because the impact of the view is restricted to the data mart, instead of the entire warehouse.
(2) Natural keys between domains can be identified and used to relate concepts in different domains, reducing the potential for undesired cross-domain interaction while supporting the desired relationships.
(3) Data-marts may be implemented using different technologies. This allows appropriate support of multi-modal data (instead of relying on a single technology such relational databases). For example, a text extraction data-mart could be used to represent documents within the warehouse.
(4) Data-marts provide an obvious partitioning upon which to distribute the data across multiple machines if the workload becomes too large for a single server to support.

Supported Queries

Our infrastructure is designed to support four distinct types of queries:

Type 1: Queries that can be answered directly by information in the warehouse. Example: list all chemicals with melting points between 57 and 60 degrees Celsius

Type 2: Queries about specific facts with specific desired answers.
Example: find all threats with delivery mechanism "salad bar" and symptom "stomach ache"

Type 3: Discovery queries where general information is required about a specific fact.
Example: Tell me everything about delivery mechanism "salad bar"

Type 4: Discovery queries where general information is required about relationships between several facts.
Example: Tell me everything about delivery mechanism "salad bar" as it relates to symptom "stomach ache"

The *fact-based queries* (Types 1 and 2) are used to identify specific information of a known type. These queries are straightforward questions that obtain direct answers. The *discovery queries* (Types 3 and 4), in contrast, are explorations of the relationship space. The analogy we use is tossing a single rock into a pond (Type 3) to indicate seeing what connects to a single starting point ("What do we know about Mr. XYZ?") and tossing several rocks into a pond (Type 4) to see how their ripples intersect ("Can we connect any group of people with an outbreak of X in Yugoslavia in 1979 and an outbreak of Y in Uganda in 1984 and with any vendors in Europe of chemicals used to process X and Y?"). We anticipate that Type 3 and Type 4 queries (as well as a subset of Type 2 queries) will be processed through the semantic network.

Data Exploration Overview

The data exploration component of the infrastructure is responsible for actually executing the queries requested by our users and displaying the results. As shown in Figure 3, there are several logical steps that need to be performed for this to occur.

When a query is submitted through one of the user interfaces, it is first transformed into a query that the warehouse understands then it is executed against the appropriate data within the warehouse. The complexity of these steps varies dramatically based on both the type of query being executed and the kind of information required to answer the query.

Type 1 queries can be executed directly against the appropriate data-mart within the warehouse. They deal directly with concepts and relationships identified and defined within the warehouse. The other types of queries are originally formulated using the concepts and relationships defined by the high-level abstractions and thus must be transformed to the query the appropriate data-marts. Because these queries may span several application domains, executing them requires the ability to navigate the semantic network, which combines the results from individual data-marts at a highly-abstracted level. They may also involve more complex analysis routines whose execution must be appropriately coordinated with the retrieval of information from the warehouse (to retrieve requested details that are not stored at the semantic network level).

Finally, once the query results have been obtained, this data may need to be transformed before being returned to the user. This transformation increases the usability of the results by both formatting them to be displayed in an intuitive way and, where appropriate, allowing them to be used as the input to a future query. While formatting the results could be as simple as displaying them in a table, it could also be complex operations such as creating histograms, mapping the results to images, or translating them to the format required by a specific visualization tool. Creating complex transformations can require a significant effort, however, the expectation is that the resulting display becomes increasingly valuable to the end-user. Queries that apply against the

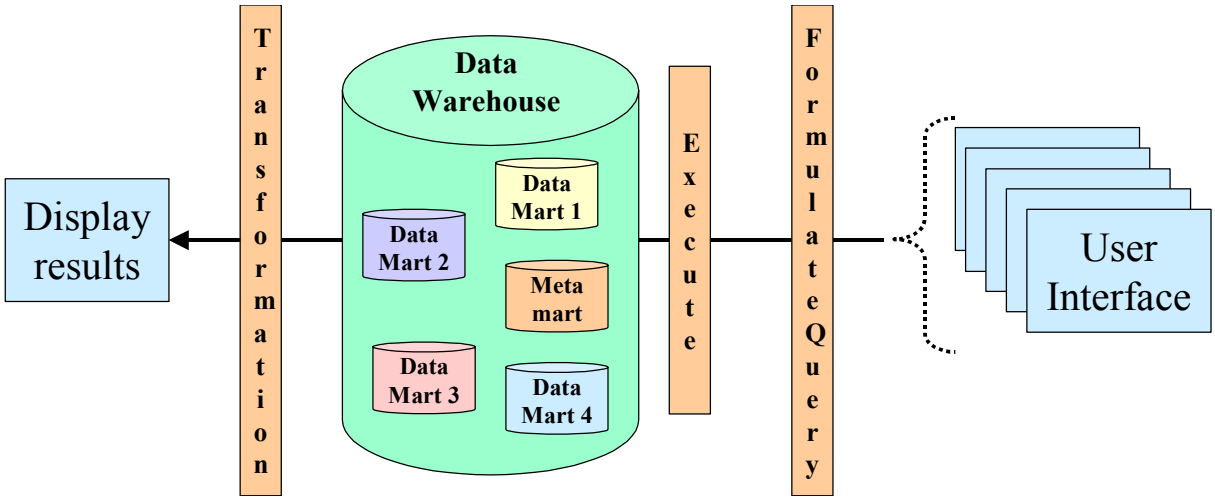

Figure 3 Data exploration functionality

semantic network will require visual navigation tools that allow the user to iteratively explore link results.

Design Considerations

There are many alternative approaches that could be used to develop an infrastructure for data integration and exploration, with the best one being heavily dependent on the detailed goals and constraints imposed on the system. This section explicitly identifies the goals, assumptions, and constraints that we have operated under to define our specific approach. It is important to realize these do not comprise formal requirements. While some of these considerations may be changed without impacting the appropriateness of our design, many changes, even small ones, will have a dramatic impact on whether or not our design is the best solution.

Goals and Guidelines

This section states the over-arching goals the system should meet. At the highest level, we have goals related to users, the system, and system maintenance. Each goal has a reason for inclusion and associated guidelines outlining how we expect it to be met.

Goals related to users

Close customer interaction **-** is essential for specifying additional requirements and functionality of the system, hardware and software requirements, and appropriate verification measures. We intend to work closely with both NBACC and Bob Burleson's semantic network team to ensure that our design continues to meet their needs. For example, the exact specification of data warehouse "feeds" to the semantic network must be carefully coordinated to ensure that biological relevance is maintained, while allowing maximal semantic linkage.

Extensive curation capabilities **-** will facilitate the data cleaning and annotation required to ensure the warehouse contains data of the highest possible quality. We will provide web-based interfaces that support curation in addition to those supporting queries. Once the system passes the early prototype stage, supporting the curators will become a major task in itself.

Intuitive interfaces - allow the user to focus on understanding the information contained within the warehouse rather than the mechanics of getting the desired information out of the infrastructure. We expect to work closely with our customers and provide online documentation to meet this goal. Related goals are to have:

Web-based interfaces - offer a high degree of flexibility and portability because they do not impose significant restrictions on the user's environment – only requiring a standard web browser be available. We intend to utilize static and dynamically generated html interfaces as well as applet / servlet interfaces to provide the most flexible interfaces possible.

Iterative query capabilities - support interactive data exploration. We intend to leverage the high-level abstractions to ensure that, where appropriate, users are able to interactively explore the data contained within the warehouse.

Complex query and result display capabilities – support discovery based queries and execution of external analysis tools. This support is important to ensure that our customers will be able to ask the questions they need to ask and have the results displayed in the format that is most important to them. We have explicitly included these steps in our infrastructure design. Building appropriate and intuitive user tools for navigating the semantic network will be a major component of the ultimate success of this system.

Goals related to the system

Globally consistent data representation – relieves the user from the burden of transforming the data from a variety of data sources to a unified view. This is especially valuable in an environment where any given user may not be familiar with all sources integrated into the system. The data will be converted to a semantically and syntactically consistent format before it is loaded into the warehouse.

Multi-modal data support – is required to ensure the warehouse is able to hold all relevant information. By using a collection of data-marts instead of a monolithic warehouse, we can develop data-marts specifically suited for multi-modal data sets such as text, image, and video data as well as traditional data formats.

Data provenance tracking – is required to ensure that users are able to identify both where data came from and what transformations, if any, were applied to it. This information will be explicitly stored in the data warehouse, and all of our data transformation, ingest, and curation tools will update the data provenance as appropriate. The semantic network system must similarly be able to link back to the underlying data warehouse facts.

Generalized software components - allow the infrastructure we develop to be generally applicable to a wide variety of application domains. While domain specific components, such as specialized query interfaces, will exist, whenever possible we will use good software engineering techniques to develop generalized components by isolating the domain specific aspects of the code.

.

Acceptable performance and security **–** are requirements for anyone to actually use the system. We will apply appropriate software engineering and performance management techniques to ensure that the system is acceptable to our customers.

Goals related to system maintenance

Tools that reduce the effort required to integrate new data sources - are required because the cost of integrating new sources into a warehouse is one of the primary reasons warehousing efforts fail. Given we are applying our infrastructure to dynamic domains where new data sources are expected to be added regularly, it is critical this cost be reduced. We are developing a suite of utilities that will reduce the cost of integrating new data sources into the system.

Tools that reduce the effort required to integrate new analysis capabilities **–**will ensure that the infrastructure continues to meet customer requirements as new analysis tools are developed and new versions of old tools are released. We are making extensive use of meta-data and using existing standards to minimize the effort required to incorporate new tools into the infrastructure.

Assumptions and Dependencies

We are currently working under a large number of assumptions, resulting in part from our inability to have significant, direct interactions with our target customer. As this work progresses and we build stronger customer ties, we expect these assumptions will be refined to better reflect the customer's true environment.

In the meantime, our prototype is being developed based on the assumptions we will:

- Work closely with bob Burleson's semantic network team.
- Tom Slezak will serve as our primary contact with our target customers at NBACC and mediator with the semantic network team.
- Support biological/chemical threat related data.
- Use the Oracle® relational data management system to store traditional, relational information within the data warehouse.
- Be allowed to make local copies of the original data.
- Use only publicly available data sources.

In addition, we have made the following assumptions when creating this design:

- When deployed, the system will reside behind a firewall.
- The data warehouse will not contain classified information.
- The end-user has access to the system via local[1] network access.
- The system will not be released as open source.
- We will not use third party software products that restrict the dissemination of our infrastructure to a variety of agencies, require our infrastructure to be released to others, or require our infrastructure to be released open source.
- Agencies utilizing our infrastructure will be willing to purchase Oracle as well as any commercial software packages required to either perform application specific queries or display the results in a desired format.

[1] "*local"* in this context refers to the network behind the firewall. The connection may be either direct or through VPN

- Existing technology can be applied to manage and query non-traditional data types.

General Constraints

This section describes global limitations having a significant impact on the design of our infrastructure. Each item listed details the general constraint with the associated impact on the system.

Constraint: *Oracle® supports traditional and non-traditional data types, however, it has only limited capabilities for queries over non-traditional data types.*
Impact: Design/development of a storage/query capability for non-traditional data types is required to allow the warehouse to function seamlessly, with no impact to the user, when querying over both traditional and non-traditional data types. The warehouse must be capable of storing both large collections of data and large instances of data objects.

Constraint: *The end-user will interact with the system over the web.*
Impact: There are multiple web browsers on the market. The design, development and testing of the system must support use of the top three most commonly used web browsers: *Netscape 7.0, Mozilla 1.5 (5.0), and Internet Explorer 6.0*.

Constraint: *The system must support common end-user analysis of data and results*.
Impact: The customer required analysis tool(s) will likely have a significant impact on both the front- and back-end components of the system (e.g., how results are post-processed for format or display; what the interface to the system looks like for analysis.) Thus the system must be designed to maximize interoperability.

Constraint: *The database must remain highly available, and not subject to a single point of failure.*
Impact: The design of the system must incorporate fault tolerance measures to ensure availability in terms of access to the data, hardware and software redundancy, and load on the server side. The system must accommodate clients with limited resources so that resultant data is displayed, processed and operated on without crashing, freezing or hindering the user. Minimum hardware/software requirements will be itemized in the appendices. We likely will initially use a development system for the initial prototype that does not support full fault tolerance. Upon putting the first version into production use, a production machine that meets the desired risk / cost balance will be procured.

Constraint: *The warehouse must provide the best quality data possible.*
Impact: The system must inherently support *curators – i.e.* people who validate, update, and annotate the data within their areas of expertise.

Constraint: *Some data sources are inherently volatile (in content, format, etc.)*
Impact: Maintaining the system inherently requires an administrator with programming expertise to manually respond to source changes when automatic techniques are insufficient.

Constraint: *The system must use appropriate standards, where possible.*
Impact: The system designers must be aware of, or able to identify standards across a variety of

relevant areas. In addition, they must be able to make appropriate choices in an environment where multiple standards or versions of a standard(s) are used.

Constraint: *Integration of new data sources requires a person to appropriately reconcile semantic differences and integrate the source into the system.*
Impact: Integrating a new data source will require someone familiar with the infrastructure interacting with a domain expert familiar with the data source and the global schema to resolve semantic conflicts. Some data will simply be in conflict over "facts", and this conflict should be represented within the system.

Constraint: *Customer specified security requirements must be met.*
Impact: The components of the system must be designed with the flexibility to add, modify, or delete levels of security. The security component(s) must be flexible enough to accommodate all customer specified security requirements.

Constraint: *Queries will not be issued outside of the local network*
Impact: Enough data must be stored on local computers to answer all user specified queries.

Constraint: *Client systems may not be able to process query results (there may be memory limitations, missing software, a low bandwidth connection, etc.).*
Impact: Where possible, the system should dynamically deal with client side restrictions and return results in a way that does not hinder, freeze, or crash the client (for example, by sending an appropriate error message to the client).

Constraint: *Customer specific performance requirements must be incorporated.*
Impact: Although this impact cannot be detailed, we will work with the customer to realize achievable performance requirements in the design, development and deployment of the system.

Constraint: *Established software engineering practices with respect to the testing, verification, and validation of the system must be adhered to.*
Impact: The system's development will include applying standard testing, verification and validation tools and techniques. Additional practices, tests, and techniques based on customer requirements will also be incorporated.

System Architecture

Section 0 presented a high-level overview of our infrastructure including the required capabilities of both the data integration and data exploration components. In this section, we present the detailed system architecture that will be used to provide these capabilities. We do not define the interfaces between the modules or other implementation details – those will be included in a future appendix to this document.

Within each component, the architecture is split into two "tracks". The first is the set of modules that make up the static view of the system. These modules, along with their associated control and data flows, are used by the system as it loads data into the warehouse or executes queries over that data. The second is the set of utilities (tools) that are used to update or extend the system when something changes. These utilities (shown as octagons in the following diagrams) are used to generate new modules or update / extend existing ones. Strictly speaking, this second track is not required since a

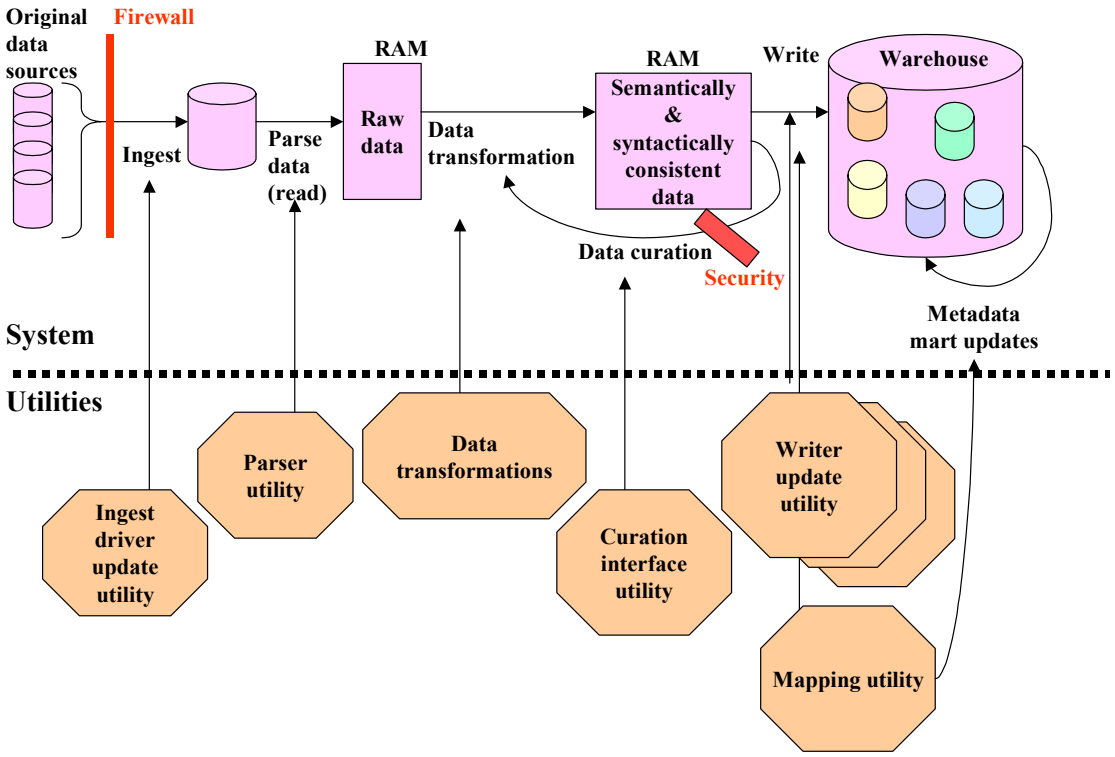

Figure 4 Data Integration Architecture

sufficiently capable administrator could perform these tasks manually. However, from a practical perspective, the existence of these tools is critical to the long-term viability of the system because they reduce the overhead required to maintain it. Because of the similarities between these utilities at the architecture level, all of the utilizes are discussed in Section 0

Data Integration Architecture

The data integration architecture, shown in Figure 4, is responsible for transferring data from the original data source to the data warehouse. The original data resides outside our local environment, and in particular, outside of the ***firewall*** protecting the system. The ***ingest*** module runs at regularly scheduled intervals. It is the routine that initiates the data transfer from the source. It contacts each source, identifies which data needs to be copied to the local system, and performs the copy. Then the data is ***parsed*** from its original format into an in-core representation, which is used as input into the data transformation step.

The ***data transformation*** step resolves semantic and syntactic differences between the source format and the global representation used by the warehouse. Performing this resolution may require extensive preprocessing of the data. The first step in automatic ***filtering*** also occurs during this step, as information that is contained in the source but is not utilized by the data warehouse is discarded. This ***filtering*** may occur prior to the data being transformed, or as part of the ***transformation*** process, and will usually involve eliminating attributes of an object that are not stored in the warehouse. Once a semantically and syntactically consistent representation of the data is obtained, a second phase of automatic ***filtering*** occurs. In this phase, objects identified as not relevant to the application domain (e.g. bioforensics) are removed. If the data is being ***curated***, it is then presented to the curator. The curator may modify some or all of the data records values, annotate the record to

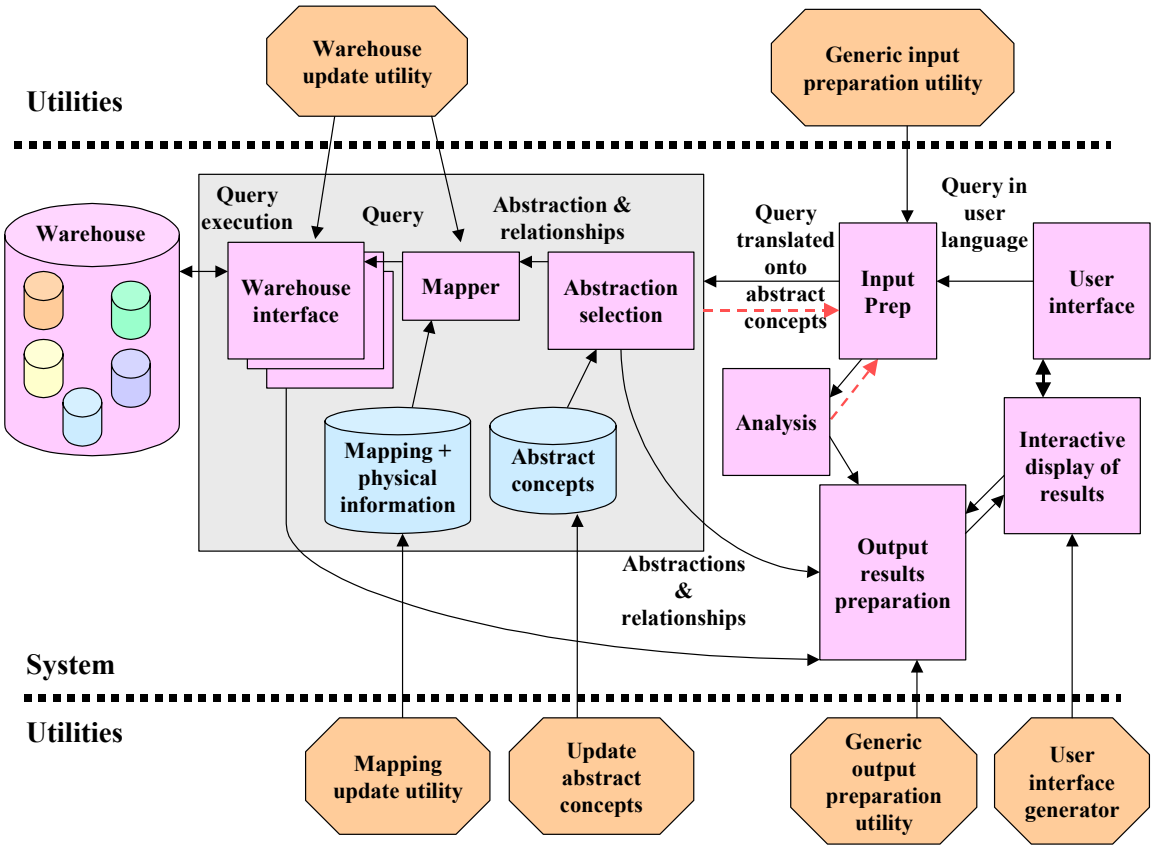

Figure 5 Data Exploration Architecture

provide additional information, or delete the entire record, preventing it from entering the warehouse. Because of the importance of the ***curation*** function, curators must log into the system and all of their changes are reflected in the data's provenance log.

The final step in the process is to ***write*** the data into the warehouse. Each ***data-mart*** contains information from a single conceptual domain (e.g. protein structures, chemical hazards, medical symptoms), however, a data source may contain information from multiple domains. Thus, the in-core data structure needs to be mapped onto the appropriate set of ***data-marts*** and the ***meta-data data-mart*** needs to be updated appropriately. Before the update is committed, a ***check*** is performed to ensure the data entry did not introduce any inconsistencies into the warehouse. Any potential inconsistencies are handles by either the curator or the system administrator.

Data Exploration Architecture

The data exploration architecture, shown in Figure 5, has both more modules and a more complex control flow than the data integration architecture. The control flow is initiated by a user submitting a query to the system through one of the ***interfaces***. The query is initially expressed in a format that the interface can easily support. That query is then converted into an executable format by the ***input preparation*** module. If answering the query requires executing an external program, the conversion translates the query to the input format required by the ***analysis application***. In most cases, however, the query is expected to be against the ***warehouse*** and only minimal conversions will be required. In order to support complex queries, the ***input preparation*** module may need to convert the input query into several steps and execute each step in order, feeding the results from one step into the input

parameters of a later step (because of the expected infrequent nature of these multi-step queries, this control flow is shown using dashed red lines in the figure).

The ***analysis*** module is responsible for executing an external analysis program on behalf of the user, and returning the results. Like the query engine module discussed next, it is expected to be a complex module. However, the details of the module are highly dependent on the tools used to perform the analysis. The selection of these tools, in turn, is dependent on the requirements of the customer. At this time, we have not had sufficient customer interaction to identify these tools. As a result, we are deferring the design of this component to a future date.

The ***query engine*** module is responsible for executing the query against the ***data warehouse*** and returning the results. Because of the types of queries being supported, this module is more complex than it might initially appear. We have decomposed the query engine into three distinct (sub-)modules. The ***abstraction selection*** module uses the query and the ***abstractions*** to select the appropriate set of concepts and relationships required to execute the query. If the ***abstractions*** are viewed as a graph, with concepts as nodes and relationships as edges, this ***selection*** can be viewed as identifying the subgraph that corresponds to the query. (Note: this is a function of the semantic network system, which must be able to link to the data warehouse for some query details.)The ***mapper*** module takes this subgraph and converts it to an executable query plan, using the ***meta-data mappings*** that relate the ***abstractions*** to their locations within different ***data-marts***. If all of the information is contained within relational data sources, the output of the ***mapper*** will be a list of SQL commands (sub-queries) that correspond to the original query. However, if the data-marts contain non-relational data, the ***mapper*** is responsible for invoking the sub-queries in the appropriate order and passing data between them as required. The ***mapper*** does not execute the sub-queries directly, instead it invokes the ***warehouse interface*** module to execute them and return the results. This module presents a consistent search and retrieve interface across all data types. In the case of relational data, this module is a thin layer that passes the requests on to the relational database system. On the other hand, each type of non-relational data contained within the warehouse has at least one customized ***warehouse interface*** module to handle that type of data. These modules support the common interface by converting the sub-query into the format required by the underlying application, running the application, and returning the results.

Once results are returned by either the query engine or the analysis program, they are ***post-processed*** before being displayed to the user. This processing converts the data from its native format in the underlying infrastructure to the format required by the user display. This ***processing*** may include operations such as filtering, aggregation, syntactic transformations, fragmenting, and sorting the data.

Finally, the ***display*** module presents the results to the user in the desired format. In general, however, these results can be used as input to another query, and thus in some sense, the display module is simply one of the supported user ***interfaces***.

Utilities

As Figure 4 and Figure 5 show, utilities play an important part in developing and maintaining our infrastructure. We anticipate having utilities to aide in the generation of each customized module as well as in the extension of module functionality. These utilities will range in complexity from simple

consistency checkers, to visual interfaces supporting meta-data updates, to automatic code generation capabilities.

Specifically, the following utilities are expected to perform consistency checks and ensure meta-data updates are propagated to all relevant locations:

- Input preparation utility
- Output preparation utility
- Update abstraction utility (feed to the semantic network system)
- Mapping update utility
- Warehouse update utility
- Curator update utility
- Writer update utility
- Mapping utility
- Ingest driver update utility

And the following utilities are expected to generate new modules and associated code based on meta-data

- Data transformation utility (generates both mediators to perform the transformations and data filters)
- User interface generation utility

Prototype Capabilities

Obviously, fully implementing this design will take significant time and effort. In order to both identify potential problems and provide initial capabilities as early as possible, we are taking an iterative approach to implementing this design. As a result, our initial prototypes will have limited capabilities and only partially implement the design described in this document.

Specifically, the prototype we will demonstrate in July will show queries running over a data warehouse combining data from three bioforensics relevant data sources into a globally consistent representation. The query interfaces will support a combination of Type 1 and Type 2 queries, including at least one query involving an external analysis tool. The sources will be integrated into an Oracle data-mart using parsers and the data transformation code produced by the mediator generation utility. The warehouse will reside entirely on one machine and ingest routines will ensure the data is regularly refreshed. No multi-modal data will be supported because all of the information from the data sources will be stored in Oracle. As a result, the mapper and warehouse interface modules will exclusively generate queries against relational databases. We do not expect this prototype to demonstrate any curation or security capabilities, any of the other utilities listed in Section 0, or any discovery query capabilities. Finally, we do not expect it to be using Bob Burleson's semantic network infrastructure to perform abstraction selection.

University of California
Lawrence Livermore National Laboratory
Technical Information Department
Livermore, CA 94551