AFRL-IF-RS-TR-2004-284
Final Technical Report
October 2004

EXTENSIBLE PROBABILISTIC REPOSITORY TECHNOLOGY (XPRT)

Alphatech Incorporated

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. P286

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

STINFO FINAL REPORT

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2004-284 has been reviewed and is approved for publication

APPROVED: /s/

CRAIG S. ANKEN
Project Engineer

FOR THE DIRECTOR: /s/

JAMES A. COLLINS, Acting Chief
Information Technology Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)

2. REPORT DATE
OCTOBER 2004

3. REPORT TYPE AND DATES COVERED
Final Feb 03 – Jun 04

4. TITLE AND SUBTITLE
EXTENSIBLE PROBABILISTIC REPOSITORY TECHNOLOGY (XPRT)

5. FUNDING NUMBERS
C - F30602-03-C-0024
PE - 62301E
PR - GENI
TA - SY
WU - S1

6. AUTHOR(S)
Fortis Barlos

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Alphatech Incorporated
6 New England Executive Park
Burlington Massachusetts 01803

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Defense Advanced Research Projects Agency
3701 North Fairfax Drive
Arlington Virginia 22203-1714

AFRL/IFTB
525 Brooks Road
Rome New York 13441-4505

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
AFRL-IF-RS-TR-2004-284

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: Craig S. Anken/IFTB/(315) 330-2074/ Craig.Anken@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 Words)*

Searching for and retrieving information from multiple, legacy data sources has become a critical need in areas such as counterterrorism data mining, all-source analysis, and law enforcement. Critical data is often scattered across conventional databases, web pages, XML (extensible markup language) repositories, and free text and multimedia data stores employing multiple schemas or even schema-less formats. The Extensible Probabilistic Repository Technology, XPRT, enables users and applications to search for and retrieve information from multiple, heterogeneous, legacy data sources. XPRT's mediators automatically translate information from source legacy formats into one or more federated schemas that mirror various application domains. XPRT's publish/subscribe service allows users to instantiate queries and have results asynchronously returned to them as soon as they become available in legacy sources.

14. SUBJECT TERMS
Information Management, Data Mediation, Counter-Terrorism Data Analysis, Link and Group Understanding

15. NUMBER OF PAGES
54

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

TABLE OF CONTENTS

LIST OF FIGURES

1 – SUMMARY

The eXtensible Probabilistic Repository Technology (XPRT) is a component of DARPA's Genisys program. The goal of Genisys was to develop the next generation database technology that will make it easier and cheaper to exploit distributed information sources and improve the ability to represent uncertainty in structured data. XPRT was a five year plan, the first year being focused mostly on research and exploitation of various technologies. The Genisys program was part of the DARPA Terrorist Information Awareness program (TIA) that was intended to prevent asymmetric attacks from terrorist organizations on US assets.

The TIA program was cancelled during its first year by Congress. As a result of this action, the scope of the XPRT program was reduced significantly in functionality, and the new focus became the implementation of a prototype system that addresses the data access needs of the Intelligence Community. We built XPRT as a middleware application that fuses information from multiple, external databases and combines this information into a single, coherent view of the requested data. Applications define their data requirements and data mapping through a domain ontology that is then used by XPRT to generate the appropriate object retrieval and fusion processes. The following figure illustrates the XPRT architecture.

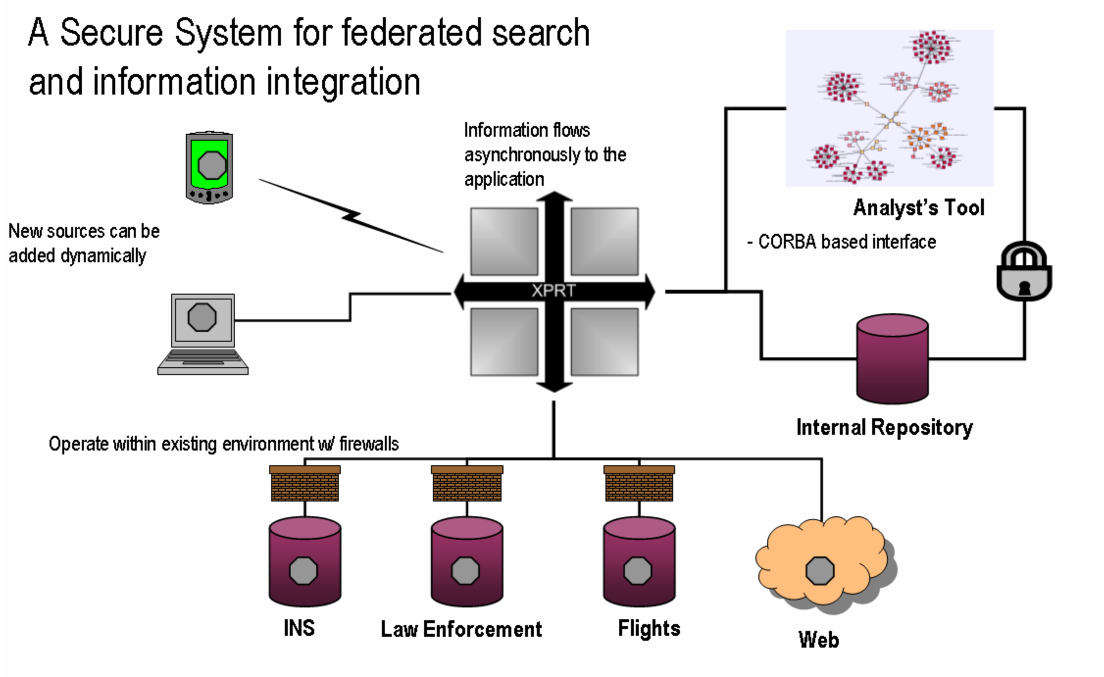

A Secure System for federated search and information integration
Information flows asynchronously to the application
New sources can be added dynamically
XPRT
Analyst's Tool
- CORBA based interface
Internal Repository
Operate within existing environment w/ firewalls
INS
Law Enforcement
Flights
Web

Figure 1: XPRT Overview

XPRT provides the following, high level, capabilities: a flexible transport mechanism to exchange objects among distributed applications, support of schema evolution for both database and in-memory objects, an object query mechanism via subscription calls, and finally, management of uncertainty in the input data.

In a typical XPRT query request, the semantic ontology of the requesting object will span multiple data sources. A single XPRT query request will result in a federated search over the various data sources, and will be handled by three independent set of processes, the XPRT service, the XPRT Aggregators and the XPRT Mediators. The XPRT service acts as an object bus and forwards objects from producer to consumer processes. The XPRT Aggregators combine individual domain objects, and the XPRT Mediators retrieve data from the data-sources.

XPRT is a distributed, loose coupled system. The XPRT processes operate independently and they all communicate via CORBA messages. With the exception of the XPRT service, that acts as the object bus, and the XPRT Aggregator, that combines the individual results into the consolidated domain object, any of the data mediators can swap in and out of processing without impacting the operation of the system; this is particularly common in federated access systems, where particular external data-sources, or the communication routes to those sources, might become unavailable independently of the other components.

The XPRT system provides two methods to initiate a search, a programmatic interface and a graphical user interface. The programmatic interface is defined by an IDL specification and is accessible via the CORBA protocol. The Graphical interface is built on top of the programmatic API and has been implemented in Java. The Graphical Interface can run as a standalone application, or it can be embedded as an applet in a Web browser.

To validate the capabilities of XPRT and demonstrate the value of information fusion in the Intelligence Community we built a specialized domain ontology and an operational flow based on a counter-terrorism CONOPS (Concept of Operations). The CONOPS involves tasks that real analysts are currently engaged in and which involve accessing multiple, heterogeneous, distributed data bases. These types of tasks currently require the analyst to manually merge data from both structured (flat file, relational, and/or object-oriented) databases and unstructured (text) databases. To simulate a real operational scenario we created several databases and we populated them with fictitious data. The data consist of millions of transactions and a corpus of several thousand documents distributed across several structured, semi-structured and unstructured databases. We built specialized mediators and aggregators based on the schema of the databases and the types of queries in the use case. Finally, we generated a GUI to issue queries and view the results.

2 – INTRODUCTION

Discovering and ultimately preventing terrorist activities requires an unprecedented collaboration among systems and human analysts to *mine* vast quantities of varied data including transactional, unstructured and semi-structured text, and multimedia data such as imagery and video, and to *monitor* these sources for suspicious patterns of behavior. These requirements were identified by Genisys, a five year technology program that set out to develop the next generation databases to allow efficient integration of data and enable collaboration between applications.

The Genisys Program Area Definition (PAD) [12] identified several limitations with the current database solutions. These limitations make the current databases technologies unsuitable for preventing terrorist's activities, and the PAD stipulated that most of the problems arise from exposing the internal database structure to users and applications. The basic approach to this problem, proposed by Genisys, was to create an abstract schema and a translation mechanism so that users and application developers only have to know one, virtual database schema that is then mapped to the physical schemas used in the actual databases. This method is implemented by software agents called mediators. Genisys also recognized that most of the information in the current systems contains uncertainty, therefore any database solution must be able to handle data with uncertain values. Finally, the database solution must be able to scale without bound with a distributed architecture.

The eXtensive Probabilistic Repository Technology, XPRT, is the technical solution to the Genisys program. It started as a five year effort, but it was later reduced in scope to a prototype program scheduled to be implemented into one year. As a result of this reduction in scope, XPRT grew into a middleware technology capable of efficiently storing and rapidly accessing large quantities of heterogeneous data, handling information uncertainty, and monitoring a wide variety of data sources for new, relevant data as soon as it becomes available. Figure 2, below, illustrates the components of the prototype system as they relate to the full program. Greyed-out components were not implemented in the prototype. Most of the other components of the full Genisys program were implemented in the prototype, with the exception of the Privacy Services, the Dynamic Agents, and the Ontology Manager.

More specifically, the XPRT prototype consists of the following components:

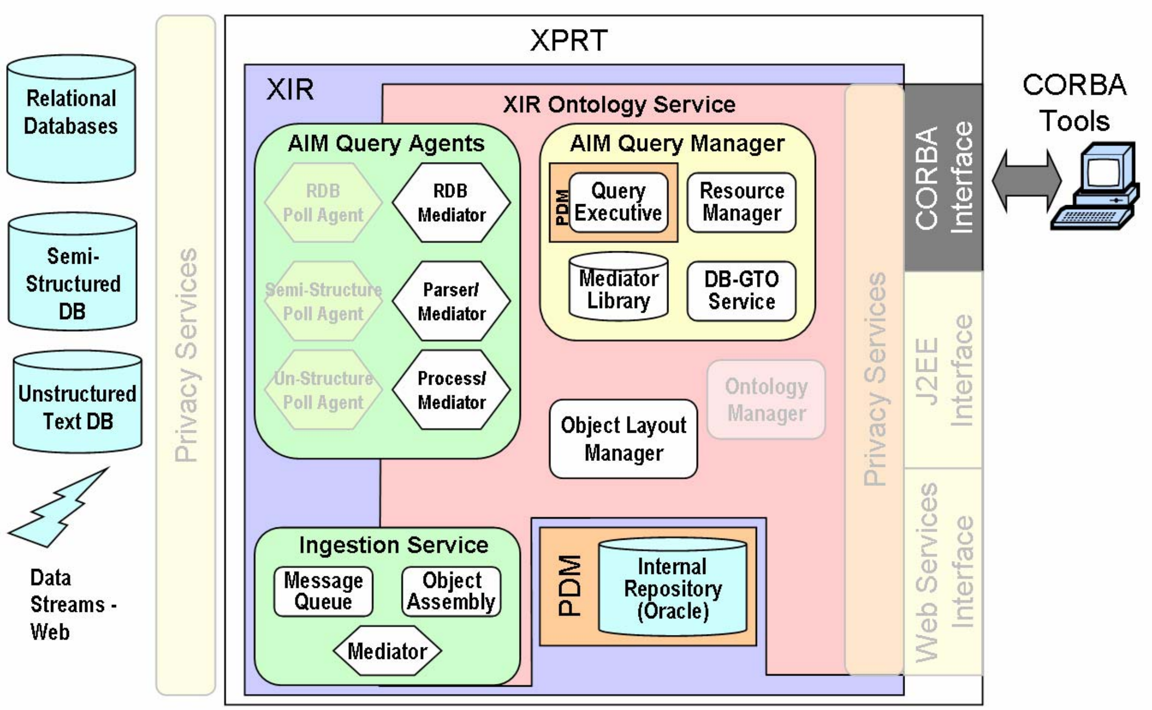


Figure 2: XPRT System

- **Query Manager:** This component controls the processing of user queries to the XPRT System. Specifically, it determines if the query can be satisfied by data in the XPRT Internal Repository or by data that has to be pulled and mediated from External Data Source(s). User queries are object-based queries expressed in the eXtensible Distributed Architecture (XDA) query language. In the test system, the Query Executive can either task the Mediator to pull data from an External Data Source and ingest it into the Internal Repository or query and retrieve data from the Internal Repository.
- **Mediators:** This component consists of a set of specialized processes that perform translation of both queries and data between the schema of an External Data Source and the internal ontology model maintained by the XPRT system. In the prototype system, the Mediators are processes specialized for each source and are cognizant of its schema.
- **Ingestion Service:** This is a database abstraction layer that serves as an interface between the internal Object Model and the underlying database implementation for the Internal Repository. This layer provides interfaces to persist, query, and retrieve operations on the Internal Repository. This layer also performs object-to-relational mapping between the internal Object Model and the relational schema in the Internal Repository.
- **Internal Repository:** This is the internal persistent cache for the XPRT System. It is an Oracle 9i database, and the schema of this database is *automatically* generated by the DB-GTO Service based on the internal Object Model.
- **Probabilistic Data Management (PDM):** A probabilistic query engine & associated services to manage uncertainty across the XPRT information domain. For the purposes of the prototype, PDM was used to predict the similarity of the biometric statistics between individuals.
- **Adaptive Information Management (AIM):** The original intent of the AIM was to provide an active data management framework for agent-based push and pull data access, to support distributed query management across a wide range of data sources, and to

utilize a novel adaptive planning technique to optimize system performance. For the purposes of the prototype, the dynamic agents were replaced by domain experts, which are individuals that understand the domain problem in great detail and have experience in object modeling and object-oriented programming.
The Domain Experts develop the mediators in the Java programming language and implement the logic to transform the domain objects from the physical representation in the external databases to the domain ontology specified by the client applications.

- **Legacy Database Support:** We demonstrate integration with three types of databases, relational, semi-structured (XML-based), and unstructured (plain text reports). In addition we can ingest data from streams, such as the WEB and WEB services.
- **CORBA Interface:** Finally, out of the three proposed interfaces, CORBA, Web Services and J2EE, we only implemented the CORBA interface. In addition we provided a Graphical User Interface that non-technical users can use to access the system. The GUI interfaces with XPRT via the programmatic CORBA API.

The XPRT program, even in its current reduced form, addresses the fundamental Genisys goals of data abstraction, uncertainty and scalability in the following manner, as specified in the Genisys PAD [12].

Scalability in XPRT is achieved via three mechanisms: query parallelism, distributed processing and multi-threaded execution of query requests. A front end process decomposes complex query requests into a set of parallel execution operations. These operations are then mapped to a set of system processes that are distributed across the available hardware resources of a distributed system. Each process operates close to the data source, is cognizant of a particular object of the domain ontology and performs simple data mediation and transformation tasks. In addition to distributed execution, each XPRT process is also fully threaded. Every new query request is handled in its own thread, therefore it can execute concurrently on a multi-processor system. The result of this architectural design is that XPRT can scale evenly across SMP and MPP systems and that its processing capacity can grow “indefinitely” as the capability of the hardware systems increase over time.

Uncertainty is handled by the Probability Query Engine (PQE). PQE is a novel extension to the relational database technologies that operates over data with uncertainties. PQE resembles relational databases in several fashions; it handles data that are stored in standard relational tables and it provides an interface based on relational calculus. At the same time, it extends the relational technologies in three major ways; (a) it operates over relational tables with missing or uncertain values; (b) it automatically calculates the probability of the relational results based on the probability distribution of the input data; and (c), it provides a mechanism to rank order the results by their calculated probability. In the context of the Intelligence Applications we use PQE to score individuals based on the match of their biometric statistics to a given suspect. Since the biometric attributes are uncertain by nature, PQE assigns a probability score to each match and uses the score to rank order the results. Unlike previous ad hoc approaches for scoring individuals, the methods underlying PQE are firmly rooted in probability theory – a paradigm that has been studied for hundreds of years and that has a rich associated set of methods to avoid racial, sexual, and other kinds of discriminatory biases.

Finally, the need for an Abstract Schema is handled in part by providing a mechanism for collaborating applications of a distributed system to evolve their schema ontology without having to update the low-level transport infrastructure. XPRT supports domain model evolution without changes to the transport mechanism, by enforcing a strict separation between high-level domain definition and low-level concerns such as persistence and inter process communication. Under XPRT it is also possible to add new entities to an ontology without modifying the physical database schemas. This is accomplished through a “model-on-a-model” architecture where flexible, high-level, semantically rich domain models are encoded in terms of a fixed, generic object model for transport and persistence.

The rest of this report describes the system in more detail. Section 3, describes the CONOPS, the supporting data, and the hardware environment. Section 4 describes the technical components of XPRT. Section 5 provides the results of our experiments. Finally, Section 6 summarizes the conclusion from our work. The report closes with references and several Appendices with detailed information about the CONOPS, the domain ontology and the requirements for each component.

3 – METHODS, ASSUMPTIONS, AND PROCEDURES

The XPRT functionality was built in accordance with a use case scenario that demonstrates the needs of real-world Intelligence Agents who track the movements of specific individual suspects. To simulate the activity of suspects in a real world situation, and demonstrate the scalability of XPRT with large size data, we generated several data sources and populated these sources with a large volume of synthetic data. We also developed a domain ontology that maps to the information in these sources and customized the XPRT GUI to display the components of the ontology. Finally, we tested the prototype system in a distributed environment consisting of several Sun/Solaris machines and Windows-based client system.

This section describes the methods we followed to build the XPRT functionality and demonstrate its value.

3.1 COUNTER-TERRORISM CONOPS

The main goal of the CONOPS scenario is to identify and prioritize the specific technical capabilities that need to be developed, along with the user interfaces (UIs) and the data to drive the scenario/demonstration.

The setting for the XPRT scenario is the Terrorist Threat Integration Center (TTIC). In this scenario, we begin by focusing on an individual analyst, Jane. Jane is responsible for monitoring the activities of the terrorist group Al Qaeda. Jane has several data sources available to her. These include the traditional "HUMINT (Human Intelligence), SIGINT (Signals Intelligence), IMINT (Image Intelligence), as well as various forms of additional transactional data (e.g., travel records, immigration/customs records). In addition, Jane has a "watch list" of specific individuals that she is monitoring. Jane's watch list includes known and suspected members of al Qaeda. For design purposes, we assume that Jane's watch list contains on the order of a 100 names. For each name on the watch list, Jane may have additional information, including digital photo, eye, hair color, weight, height, date/place of birth, passport number and country, gender, known aliases, education, and known group participation and roles.

One of Jane's daily responsibilities is to monitor, update, and assess the current whereabouts and activities of the targets on her watch list, based on the available data, and provide warning about potential terrorist activities. Some of the things that Jane might be looking for include: meetings between the people on her watch list, planning/coordination activities pursuant to a terrorist attack (e.g., surveillance, bomb making), as well as day-to-day functions of a terrorist group (e.g., recruiting and training new members, financial activities). Some obvious things that Jane might look for include: more than one person on her watch list taking the same flight, more than one person on her watch list flying to the same location within some specified number of days of each other, or more than one person on her watch list staying in the same or nearby hotels within a certain period of time.

It is known that al Qaeda and other terrorist groups do not work in strict isolation from each other; rather, they regularly share techniques, tactics, and intelligence, and provide each other with access to new technologies. In some instances, these organizations will also work together and/or make local operatives available to other organizations. As a result, Jane is also interested in knowing when someone on one of her watch lists takes the same flight, travels to the same place, or stays in the same hotel as, one or more of the people on one of her colleagues' watch lists. In this scenario, we assume that Jane has access to at least a subset of her colleague's watch lists.

The scenario is structured so as to gradually illustrate the potential capabilities and technical concepts of XPRT. These include:

- Easily define queries across multiple databases without having to know the details of those databases;
- Mediate data from multiple formats into a consistent format that the analyst can use;
- Mediate data from multiple formats into the format required by analytical tools (note, this format may not be one that the analyst wants to see);
- Support drill-down and pedigree of information;
- Search over both structured and unstructured data;
- Support probabilistic queries and calculation of confidence metrics on data; and
- The ability to work with both structured and unstructured text.

– Appendix A: XPRT , describes CONOPS in detail. The scenario consists of a series of "Scenes", each of which consists of the following elements:

Scene: A narrative description of the analyst's responsibilities and what s/he is trying to accomplish;

Data: A description of possible data sources (real or synthetic, existing or to be developed);

Technology: A list of the XPRT technologies that will be illustrated in the scene;

3.2 DATA SETS

In order to support the CONOPS, above, we developed a domain ontology and a set of test data with the following characteristics:

- The domain ontology must be centered around individuals and their activities. This is to satisfy the need that we are tracking potential suspects.
- The semantic model of the domain ontology must span across multiple data sources. This is to satisfy the need that, in the real world, information is spread across multiple databases.
- There must be a mechanism to associate records between the various sources. In the context of this prototype we used the Name attribute as the foreign key across the various databases. Even though the use of the Name attribute is a very simplistic match operation, it is still used extensively by the Intelligence Community for tracking individuals. Moreover, XPRT provides the infrastructure to replace the name match with more elaborate aggregation algorithms.
- The data sets must contain uncertainty. This is to satisfy the requirement that, in the real world some information is missing, or, if it is not missing, it is known within a margin of

error – the height of an individual is between 5’6” and 5’10”. Applications that deal with real data must be able to reason with uncertain data. For the purposes of the prototype we introduced uncertainty only in the biometric statistics of individuals.

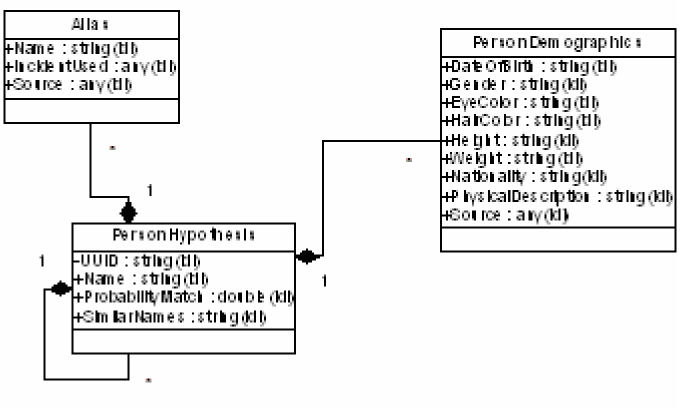
- The databases must be of reasonably large size. This is to satisfy the requirement that XPRT can mediate information from large databases. Although we designed the system with scalability in mind, we only performed experiments with gigabyte size data volumes.
- Finally, the simulated databases must represent structured, semi-structured and unstructured data models.

In order to satisfy the above requirements we created six different databases with synthetic data, including two databases from open source. Some of the data were collected from DARPA projects, such as, Centaurus, Evidence Data Base (EDB), etc., others were fabricated, such as INS and FED, while others contain data from the open source, i.e. Web, BBC, FBIS reports, etc. These data sources cover four different types of database structures, relational, XML, Web and Web-services. The following table contains information about the data sources.

Table 1: Data Sources

Data Source Name – Ontology objects		Size	Data Type
Web			
	Google Web Report	Unlimited	SOAP API
News			
	BBC News	Unlimited	WEB RSS 1.0
Centaurus			
	Person Demographics	204,402 people from 240 countries	Relational
	Flight Records	204,402 people from 240 countries	Relational
	Location Records	9 addresses	Relational
	Hotel Transactions	1,469,602 hotel records from 40,370	Relational
Simulated INS			
	Visa Applications	14 INS visas	Relational
	Points of Entry records	7,113,598 INS records	Relational
Simulated Law Enforcement			
	Aliases	33 aliases	Relational
	Incidents	46 incidents	Relational
	Human Intelligences	10 HUMINT reports	XML type
Evidence DB			
	Communication Records	1765 communication events	Relational
	Capabilities	24567 capabilities for 8783 persons	Relational
Unstructured			
	Unstructured Reports - FBIS	4110 documents	XML type

In addition, we created a domain ontology that represents the activities of individuals. The figure below illustrates a small section of the domain ontology with only three object types.



At the center of the domain model is the PersonHypothesis object. The PersonHypothesis object represents the aggregation of activities performed by individuals with the same name. Since, in the real world, names are not unique, a single instance of the PersonHypothesis object can potentially contain information about several individuals. Deconfliction of individuals must be performed programmatically by the XPRT aggregators. XPRT does not provide a build-in deconfliction engine to differentiate the individuals by their activities

In this figure we only show the PersonDemographics and the Alias objects. The full ontology contains fourteen different types of objects, such as, Hotel-Transactions, Locations, HUMINT, etc. and is shown in Appendix B

The PersonHypothesis is a self-referential object. We use this mechanism for two purposes: (a) to capture details about all the aliases of an individual, and (b), to represent information about individuals whose name matches the name of the original person. In addition, the PersonHypothesis object contains an attribute called ProbabilityMatch, which captures the distance of this individual against other persons with similar name and biometric statistics.

The sub-objects of the domain ontology map to the various simulated data-sources. For example, the PersonDemographics are stored in the Centaurus database, while the Alias information is stored in the simulated Law Enforcement database. For a complete mapping of the objects to the databases, and detailed statistics about the various databases, see Table 1, above

3.3 COMPUTER INFRASTRUCTURE ENVIRONMENT

The computer infrastructure utilized for this prototype consisted of the following hardware and software:

Hardware:

- Database Tier:
  - Server: Sun 280R
  - Operating System: Solaris 8
  - CPUs: 2, 900 MHz Sparc III processors with a total of 16MB of eCache
  - Memory: 2GB
  - Internal Disks: 2, 36GB, 10,000 RPM SCSI (Used only for operating system)
  - Network Interface: Quad-speed 100 Mb/s
  - Disk Interface: Fast-Wide SCSI III
- Application Tier:
  - Server: Sun 280R
  - Operating System: Solaris 8
  - CPUs: 2, 900 MHz Sparc III processors with a total of 16MB of eCache

Memory: 2GB
Internal Disks: 2, 36GB, 10,000 RPM SCSI
Network Interface: Quad-speed 100 Mb/s
Storage Array:
Sun StoreEdge 3310 SCSI Array
512GB cache
4, 36GB, 10,000 RPM SCSI Disks
~70GB usable space (RAID 0 + 1 configuration)
Software:
Database Tier:
Oracle 9*i* RDBMS Enterprise Edition release 9.2.0.3
(All software and database files are stored on the disk array)
Application Tier:
Oracle 9i Client release 9.2.0.3
J2SE 1.4.2
XPRT (Java components)
Mediator (Java)
(All software is stored on internal drives)

4 – TECHNICAL APPROACH

This section provides the technical details of the XPRT system. The first section describes the operational model we used to achieve scalability and high performance. The second section describes the eXtensible Distributed Architecture framework, a middleware engine for the efficient transport of binary data. XDA grew out of several AFRL programs and provides the underlying data transport and schema evolution services. The third section presents the various components of the system in more detail. Finally, the fourth section describes the services provided by XPRT.

4.1 DISTRIBUTED OPERATIONAL MODEL

XPRT mediates information from both unstructured and structured data sources and combines this information into a single, coherent view of the requested data. A client application establishes a subscription channel with the server and makes a request for data. The request is encapsulated in a query object that contains the domain ontology and the query parameters. Upon retrieval of the query request, the XPRT service initiates a ‘standing query’ operation. The ‘Standing query’ mechanism operates as an email or a bulletin board application; the service first federates the query to the various aggregators and mediators, and then it starts polling for new data. When new information becomes available, it notifies the subscribed clients to retrieve the new data. Figure 3, illustrates the various components of XPRT.

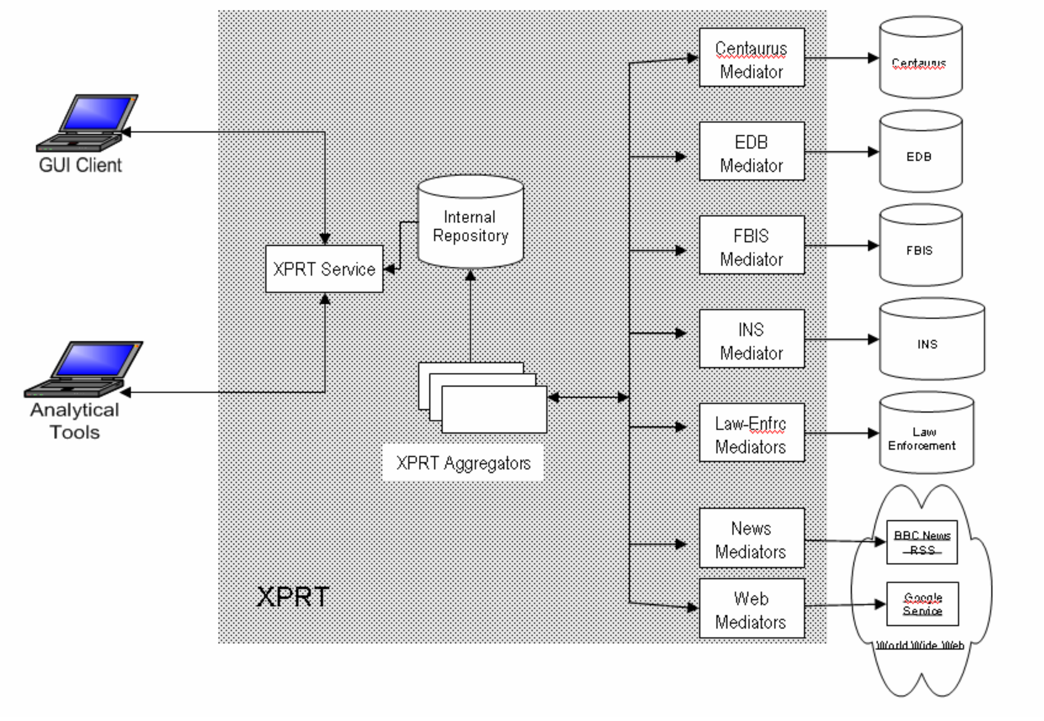


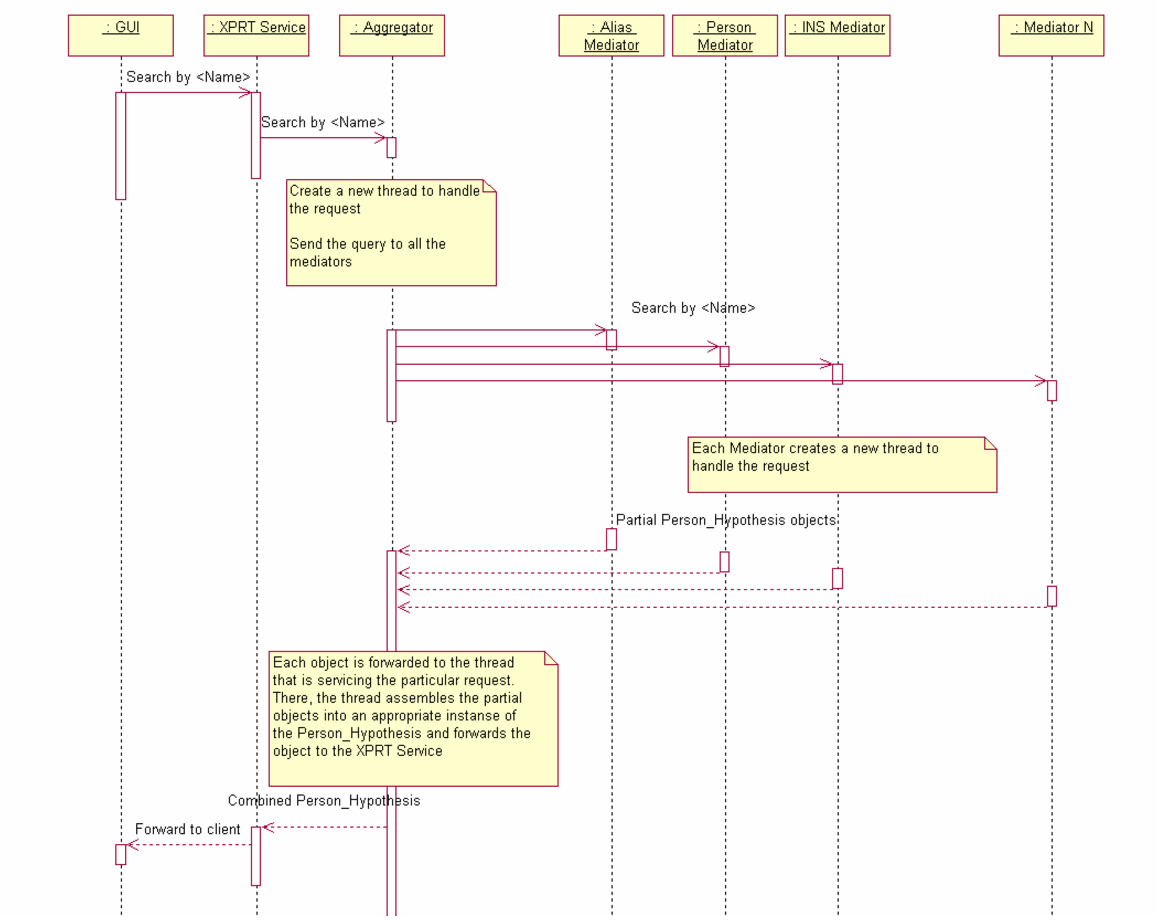
Figure 3: XPRT Architecture

The Data Access Mediators are responsible for retrieving the semantic components of the domain object. Each mediator has been coded to operate with a particular semantic component of the object. Moreover, each mediator is cognizant of a particular database schema. The Data Access Mediators query the external databases, convert the results to instances of their domain objects, and forward the resulting objects to the XPRT Aggregator. The Aggregator combines the individual objects into the full domain ontology object and persists the results in the Internal Repository.

For example, the Centaurus mediators consist of four processes that retrieve the Person Demographics, the Flight records, the Location records, and the Hotel Transactions from the Centaurus database (see Table 1 for a mapping of the ontology objects to databases).The Aggregator process receives these objects and builds the composite PersonHypothesis object.

Finally, the XPRT service retrieves the object from the repository and forwards it to the client application.

Each query request is handled in its own thread. The sequence of operations for a query execution is illustrated in Figure 4, and described below.



Create a new thread to handle the request

Send the query to all the mediators

Each Mediator creates a new thread to handle the request

Each object is forwarded to the thread that is servicing the particular request. There, the thread assembles the partial objects into an appropriate instanse of the Person_Hypothesis and forwards the object to the XPRT Service

Figure 4: Processing Model

1. The GUI makes a new query request. The request can be one of four types, Exact Name Match, Alias Search, Probabilistic Match, and Group Detection. The GUI packages the query type and the query search predicates into an XDA Generic Transport Object and sends the object to the XPRT service.

2. Upon receipt, the XPRT Service performs the following operations. It invokes the Universal Unique Identifier (UUID) generator to get a unique ID for the query, it adds the UUID in the query object, and broadcasts the query object to its aggregators via the subscription channels. It then returns the UUID to the GUI client to be used as an identification number for the results.

3. We have built four different types of Aggregators into the XPRT prototype, one for each query type, Name match, Alias, Probabilistic Match and Group Detection. To ensure that the appropriate query type is forwarded to the proper aggregator, we associate an XDA query filter with each channel. Only the query that passes the filter criteria is forwarded to the recipient process of the channel.

4. The aggregator receives and forwards queries and combines results. It operates in two major modes, the polling mode and the query service mode. The polling mode is implemented by the polling thread. This is the main thread of the Aggregator and the one that is started when the aggregator is initialized. The polling thread polls for new queries or new partial results. Once it receives a new query it spawns a new query thread to process the results, associates

the thread id to the query UUID, forwards the query to the Data Mediators, and then goes back into polling.

5. The Data Mediators operate similarly to the Aggregator. They have two basic modes of operation, polling and query service. The polling thread polls for new query requests. Once a new query request arrives, the polling thread spawns a new query service thread, associates the thread id to the query id and goes back into polling.
6. The Query service thread will periodically query the data store for new data. When new data are available, the thread creates a new partial Person_Hypothesis object with the new data, inserts the query UUID in the new object, and forwards this object to the Aggregator.
7. When the polling thread of the Aggregator receives a new partial Person_Hypothesis object, it locates the query thread for that object, and forwards the object to that thread. It then goes back into polling.
8. The Aggregator query thread receives partial objects from the polling thread, combines them into a complete Person_Hypothesis object and forwards it to the XPRT Service.
9. The XPRT service will periodically search the Internal Repository for new data and, if it finds any, it will notify the appropriate client. Once the client requests the data, the XPRT service will extract the objects from the IR and forward them to the client.

4.2 EXTENSIBLE DISTRIBUTED ARCHITECTURE (XDA)

XPRT uses the XDA framework for data abstraction, distributed process configuration and management, data transport, and database services. XDA was developed as part of several AFRL programs [14, 15, 16] to address the needs of distributed, collaborating applications.

A major difficulty in building component-based distributed systems arises from the informational dependencies between components required for inter-process communication. A key design goal of the XDA program was to build a framework that allows us to control this complexity and build flexible, maintainable, distributed systems. In particular, this technology can be used to build state-of-the-art, operational, distributed fusion systems. The overall strategy for handling the requirements mentioned above was to de-couple the high-level description of the operational domain of interest (ontology) from the low-level data transport and database persistence mechanisms (schema).

4.2.1 Model-on-a-Model Architecture

A fundamental design feature of the XDA architecture is the separation between a high-level domain definition (for example a WMD search data model) and the low-level persistence and inter-process communication details. XDA is based on the capability to build sophisticated, flexible, high-level models of the problem domain whose implementation is accomplished in terms of a fixed low-level object model called the Generic Transport Object (GTO) model.

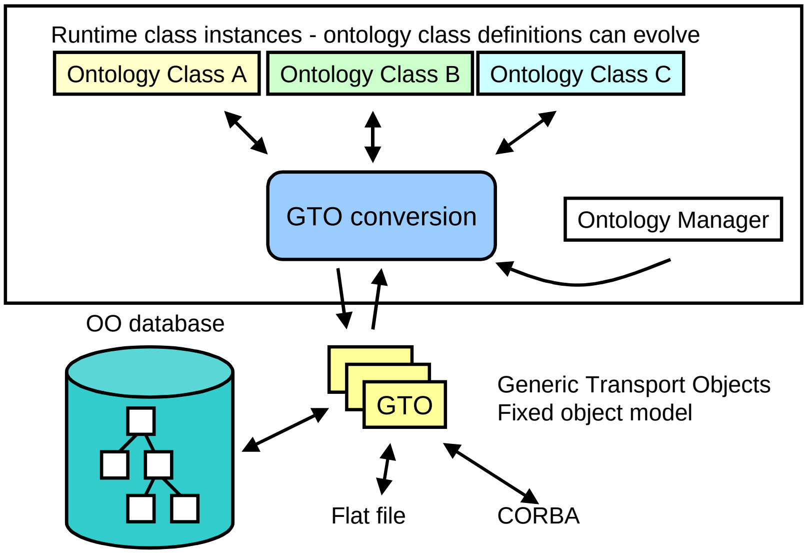


Figure 5: "Model-on-a-Model" Architecture Facilitates System Evolution

The figure above illustrates the "model-on-a-model" architecture of XDA. The runtime classes conform to a flexible (evolve-able) domain definition. At runtime we have objects that are instances of entities. Examples of these entities include target hypotheses, USMTF messages, and system messages. A set of related entities forms an "ontology." Persistence and transport utilize a fixed generic object model called the Generic Transport Format (GTO). Ontologies can be modified and can evolve without requiring modifications of the transport model. Generic Transport Objects carry both data and structure information. The containment hierarchy and the physical layout of the attributes of a class are retained when an instance of the class is converted to a GTO. Thus, the attribute layout and the containment information can be saved if an Object-Oriented database is used for persistence. This structure information combined with the ontology-based semantics can be used, for example, to implement efficient object-oriented queries.

An *ontology* consists of a set of *entities* and a set of e*numerated* types. Each entity contains a set of *attributes* that have a *name* and *data type*. An entity can be derived from a *base entity* (single inheritance) thus inheriting the base entity's attribute set. The base entity is specified by its name and by the name of its ontology. Supported attribute data types include the basic types (integer, double, and string), and sequences of those basic types. Additionally, the data type of an attribute can be *user-defined* or *user-defined-sequence*. When an attribute is *user-defined* its value is an instance of an entity that has been defined either within the same ontology or in a different ontology. Similarly, an attribute with type *user-defined-sequence* has as a value that is a sequence of instances of user-defined entities. Finally, the data type of an attribute could also be a *user defined enumerated type*. Consequently, this mechanism encourages the creation of a set of definitions of domain entities and concepts that can be incrementally refined and extended through inheritance and containment. Our use of the term *ontology* is therefore highly restricted and should not be confused with more complex constructs such as knowledge bases.

The ontology approach allows users to generate high-level descriptions of the entities of interest in the problem domain. Once an entity has an ontology-based definition, it can be converted to a Generic Transport Object (GTO) and instances of this entity can be persisted to a database, saved to a file, or transported across platforms as byte-streams.

4.3 COMPONENTS

The Distributed Operational Model and the XDA transport, described above, provide the substrate services that offer efficient query execution and data schema independence. The XPRT components are then built on top of these services to take advantage of these capabilities. XPRT consists of four major components: (a) the XPRT service, that manages the subscription/publish channels between the producer applications and the consumer clients, (b) the Internal Repository that store the temporary results and makes them available to multiple clients, (c) the various Data Mediators and the Fusion Aggregators, and (d), the Client GUI. The sections below describe these components in more details.

4.3.1 XPRT Service

The primary purpose of the XPRT Service is to provide a programmatic interface for clients to issue federated queries and retrieve the results. Input queries are cached for reuse; other clients can connect to a running query and receive the same results. The functionality of the XPRT Service is immutable to new domain ontologies and data sources.

Processes subscribe to the XPRT Service to query the databases. In addition, processes can also connect to the Service to publish results. The XPRT service operates as an object bus. It maintains a mapping of publishers, that produce data, to subscribers, that request the data, and it channels the incoming data to the appropriate processes that have requested them.

All queries that have been issued to the Service operate in standing mode; periodically the Service will receive new data from its publishers, or the Internal Repository, and will forward the data to the clients that subscribed for them. Since the results are incremental, they will need to be cumulated by the client.

The XPRT Service is a task-able XDA component. This means that it is managed by the XDA task manager and that the topology of the whole XPRT system is controlled by the XDA task topology. The topology of the system includes the Service, the Aggregators and the Data Mediators. This design also leverages XDA in defining the publish/subscribe links between all these components.

The XPRT service provides a CORBA based programmatic API to issue queries. The interface was designed with simplicity in mind. We provide methods to connect to the service, issue queries, check for new data and retrieve new information. For a complete description of the XPRT service API, as well as, all the XPRT and XDA available classes see the javadocs in the distribution folder of XPRT[1].

4.3.2 Internal Repository

The Internal Repository (IR) serves as a cache for the data gathered by the aggregators. The IR is implemented in Oracle 9i and uses the database layer of XDA to save and retrieve records from the database. Use of the IR follows the pattern below:

- When a client issues a new query, the XPRT Service will first check its internal list of standing queries. If there is a match, which means that the query is already running, the XPRT Service will extract the data from the IR and send them to the client.
- If there is no match, then the XPRT Service will publish the query to the aggregators and add the query to its list of standing queries. It will then poll the IR for new data.

[1] Javadocs are not included in the distribution file but they can be generating by executing the following command from the root folder of the XPRT distribution:

```
> ant javadocs
```

- In response to the query request, the Aggregators will persist Person_Hypothesis objects in the Internal Repository

The following diagram illustrates the operational flow of the Internal Repository

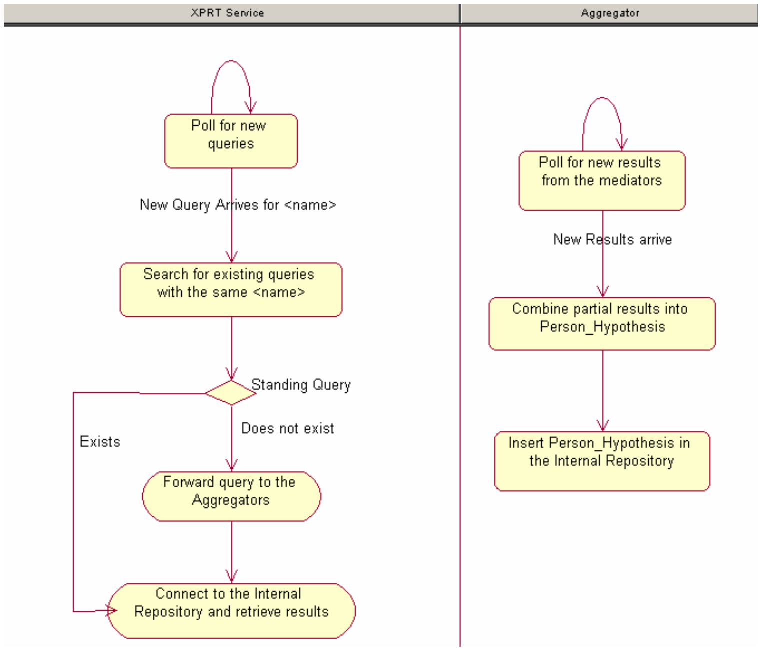

XPRT Service
Aggregator
Poll for new queries
New Query Arrives for <name>
Search for existing queries with the same <name>
Standing Query
Exists
Does not exist
Forward query to the Aggregators
Connect to the Internal Repository and retrieve results
Poll for new results from the mediators
New Results arrive
Combine partial results into Person_Hypothesis
Insert Person_Hypothesis in the Internal Repository

Figure 6: Internal Repository

4.3.3 XPRT Aggregators

The XPRT Aggregators fuse partial data from the mediators. More specifically, they perform the operations described below:

- Receive a Query Object from the XPRT Service via an XDA input channel
- Spawn a new thread that will:
    - Publish the Query Object to its subscribers
    - Retrieve domain ontology objects from its subscribers
    - Combine these objects into the PersonHypothesis object
- Persist the Person_Hypothesis object in the Internal Repository.
- Close a specific query thread in response to a client request.

We have developed five types of aggregators: Name aggregator, Alias Aggregator, Probabilistic Query Engine (PQE) aggregator, Unstructured Data aggregator, and Group Detection aggregator. The Name aggregator simply combines objects returned from the mediators. The Alias aggregator retrieves the known alias for a specific individual, and, for each alias, it issues a separate name query. The PQE aggregator first finds all the persons whose name sounds like the input argument, and for each such person, it uses PQE to compute the distance between his/her biometric and those of the input individual. The Unstructured Aggregator searches a corpus of unstructured documents for reports that match the input arguments. Finally, the Group Detection aggregator executes the Group Detection Algorithm (GDA) [19] and forwards the results of GDA to the client.

4.3.4 XPRT Mediators

Mediators are defined as software modules that interact with a user and a variety of data sources to provide one-stop shopping for an organization’s data [20]. This approach is especially attractive when taken over a large, heterogeneous system of data, because the cost of source integration for such a system is high. Intranet and digital libraries could benefit from mediators because they handle myriad data types and must treat all data appropriately, according to its type.

It has been proposed in the literature [2] that mediators perform both the data access operations and the information fusion processes. Also, some researchers have proposed intelligent mediators that adapt to the underlined schema of the external database on demand. In the context of the XPRT prototype, however, the mediators perform the data retrieval operations, only. The fusion of information is performed by the Aggregators.

Mediators retrieve data from the external data sources and return the data to the aggregators. The mediator process receives the query objects from the aggregators, and, for each such object, it spawns a new thread to query the database and retrieve the relevant data that satisfy the specific query predicates. It then converts the data source results to the appropriate object of the domain ontology and forwards the objects to its aggregators.

Mediators are the components of the XPRT system that are specific to the particular domain ontology and problem. Domain Experts must build new Mediators for each new domain problem, and implement the database access and data transformation logic in these components. The XPRT system provides a code generator that creates skeleton mediator classes to be used as the basis for implementing the domain specific logic. These base classes provide three baseline capabilities, the transport mechanisms for the transfer of the queries and the result data, the multi-thread support for managing each query in a separate thread, and the connection polling for the re-use of the database sessions.

In addition to query objects and results, Mediators also process actions originating from the GUI that are passed to them through the aggregators. In this implementation of the XPRT prototype, the only action that has been implemented is the query stop action. When the mediators receive this action, they stop the thread that processes the specific query.

The diagram bellow illustrates the operation of the mediators:

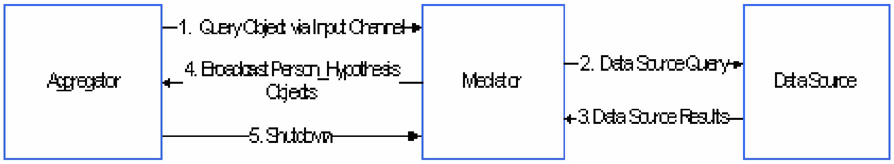


Figure 7: Aggregator/Mediator Interaction

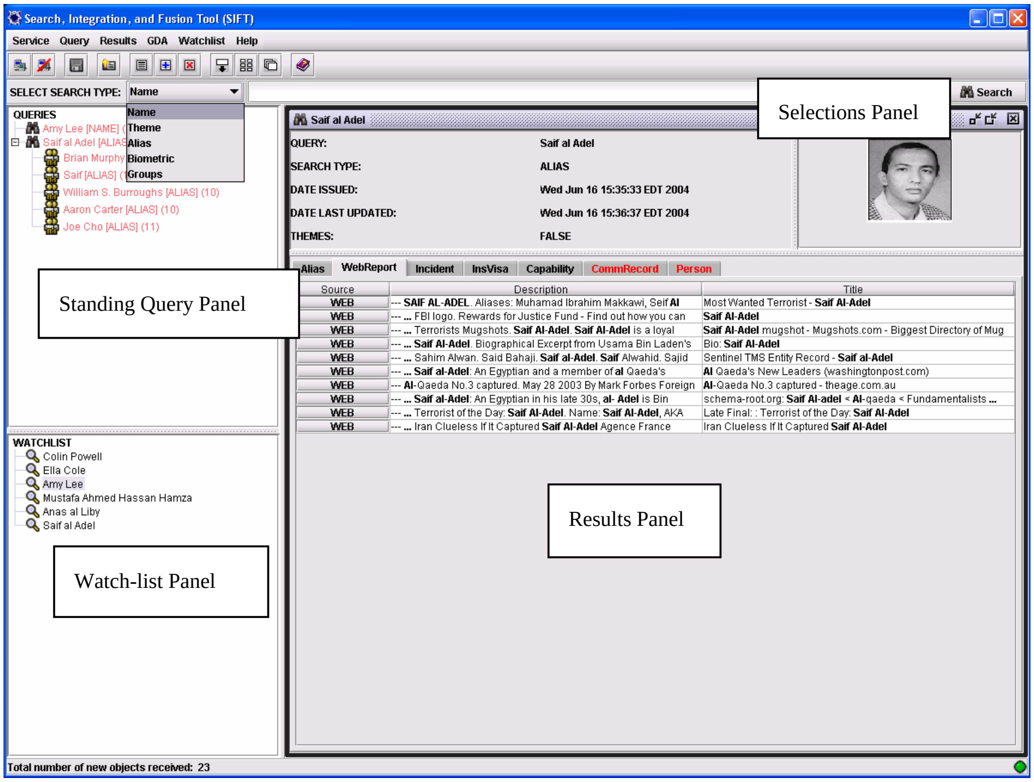
4.3.5 Client GUI

In addition to the programmatic API, XPRT provides a client GUI to issue query requests and retrieve information about individuals from the multiple data sources [18]. The GUI is written in Java and it interfaces with the XPRT service via the Service’s programmatic API. The GUI provides the following high level capabilities:

- The GUI displays results in a streaming fashion. It notifies the user every time new data is available, and, after the user has retrieved the data, it will provide visual cues to differentiate the new information from the old one.
- The results are displayed in a tabular manner; one tab per ontology object.

- The layout of the results is automatically generated from the domain ontology.
- The GUI displays results incrementally to the user.

The overall layout of the GUI is displayed in the following figure.



QUERY: Saif al Adel
SEARCH TYPE: ALIAS
DATE ISSUED: Wed Jun 16 15:35:33 EDT 2004
DATE LAST UPDATED: Wed Jun 16 15:36:37 EDT 2004
THEMES: FALSE

Source	Description	Title
WEB	--- **SAIF AL-ADEL**. Aliases: Muhamad Ibrahim Makkawi, Seif **Al**	Most Wanted Terrorist - **Saif Al-Adel**
WEB	--- **...** FBI logo. Rewards for Justice Fund - Find out how you can	**Saif Al-Adel**
WEB	--- **...** Terrorists Mugshots. **Saif Al-Adel**. **Saif Al-Adel** is a loyal	**Saif Al-Adel** mugshot - Mugshots.com - Biggest Directory of Mug
WEB	--- **... Saif Al-Adel**. Biographical Excerpt from Usama Bin Laden's	Bio: **Saif Al-Adel**
WEB	--- **...** Sahim Alwan. Said Bahaji. **Saif al-Adel**. **Saif** Alwahid. Sajid	Sentinel TMS Entity Record - **Saif al-Adel**
WEB	--- **... Saif al-Adel**: An Egyptian and a member of **al** Qaeda's	**Al** Qaeda's New Leaders (washingtonpost.com)
WEB	--- **Al**-Qaeda No.3 captured. May 28 2003 By Mark Forbes Foreign	**Al**-Qaeda No.3 captured - theage.com.au
WEB	--- **... Saif al-Adel**: An Egyptian in his late 30s, **al- Adel** is Bin	schema-root.org: **Saif Al-adel** < **Al**-qaeda < Fundamentalists **...**
WEB	--- **...** Terrorist of the Day: **Saif Al-Adel**. Name: **Saif Al-Adel**, AKA	Late Final: : Terrorist of the Day: **Saif Al-Adel**
WEB	--- **...** Iran Clueless If It Captured **Saif Al-Adel** Agence France	Iran Clueless If It Captured **Saif Al-Adel**

Selections Panel

Standing Query Panel

Results Panel

Watch-list Panel

Figure 8: GUI Layout

The GUI is divided into four major panels, selections panel, standing query panel, watch-list panel and results panel. To start a search the user enters a name in the selection panel and chooses a particular query type. XPRT supports five types of queries, name, alias, theme, biometric match and group detection. See 4.3.3 for a description on each query type.

A query that is submitted for execution gets added in the 'Standing Query' panel (SQP). This panel contains the queries that the user is currently monitoring. Users select queries from the SQP to view the results. Queries that have new data are highlighted in red. When the user has viewed the results, then the query entry turns green.

Users manage their lists of suspects in the watch-list panel. This panel provides the mechanisms to insert and delete names, save the watch-list in a file, and select individuals for a search.

The results from a search are displayed in the Results Panel, one tab per object of the domain ontology. New information is highlighted in red, and information that has been viewed is shown in black. In the example shown in the above picture, the GUI displays the information for one query request, 'Saif Al Adel'. Each tab in the Results Panel presents the data from the corresponding object types. The WebReports panel, for example, lists the entries from the Web-Google data source. The first five panels are rendered in black, which indicates that these data have been viewed by the user. The next two panels, CommRecords and Person, are highlighted in red, which indicates that this is new information. New information remains highlighted in red until the user refreshes the tab with new data from the server. When that occurs, all existing records turn into black, and any new records are highlighted in red. New data highlights apply to records, not only object panels. Thus, for example, if some new communication record becomes available, then the record will appear as red in the CommRecords panel.

The format of the Results Panel is similar for all the query types, except the Group Detection. The Group Detection Query clusters individuals based on ‘similar’ activities and this information is better represented by a graph. In the XPRT prototype we have integrated GDA with the Evidence DB database (EDB) only. The EDB database represents ‘similar’ activities as those activities that share the same EDB link. For more information on the EDB schema and the semantics of the links see [21]. The results from the GDA execution are displayed as a graph with nodes representing individuals and edges representing inferred associations between the individuals. The figure below, Figure 9, shows the result from a Groups type query.

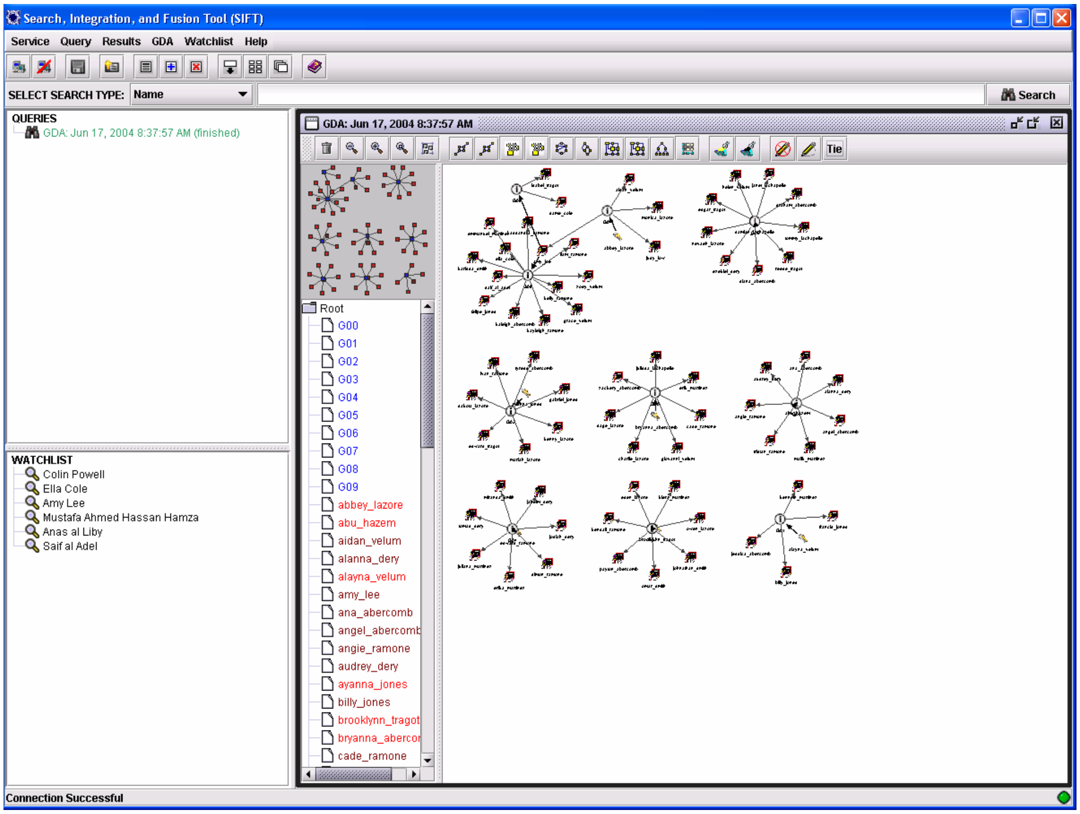


Figure 9: Visualization of Group Detection results

The figure above displays the clustering results of GDA based on the link type of Directing Agent. In this particular example we searched for 10 groups. The graph panel shows the individuals that were found to be associated, and the group they are associated with. The graph panel provides several layout options, zooming capabilities, and search.

Standing queries are maintained by the XPRT service; the GUI does not store any state other than the results of the selected query which are displayed in the result tabs. Users can get a list of the standing queries that are currently under execution by the server and then select queries to attach to. Since the results of the standing queries are maintained by the server in the Internal Repository, attaching to a standing query does not initiate a new search; the server simply retrieves the results directly from the IR and forwards them to the GUI. Users can let queries continue to execute, disconnect from the GUI, reconnect at a later time and attach to a previous submitted query to see both old and new data. Multiple GUI clients can attach to the same standing query.

Since the list of queries, and the results for each query, is maintained by the server, the access to this information can be managed by an accredited Multi-Level Security (MLS) process. Although we did not implement MLS in this prototype, we provide the architecture to easily incorporate MLS policies and restrict the access of the data to individuals that have the appropriate classification and group clearances.

4.4 SERVICES

XPRT provides five types of services, (a) exact name query, (b) alias query, (c) probabilistic match query, (d) unstructured data search, and (e) integration with GDA. The sections below describe these services in more detail.

4.4.1 Name Search

The purpose of this search is to track individuals based on their exact name. When the Name Search is issued, XPRT retrieves information from the external databases about individuals whose name matches exactly the term specified in the query search. Each Name query operates as a standing query request. The clients don't need to issue subsequent queries in order to get new data for the same individual; as new information becomes available in the external databases XPRT will collect it and forward it to the GUI.

Although the Name Search is a useful process to track individuals, it will miss misspells or different spellings across the various data sources. These specific capabilities are addressed by the Alias and the Biometric search queries that are described below.

4.4.2 Alias Search

One of the major requirements in the Counter-terrorist community is to retrieve information for an individual based on his/her primary name and all his/her known name aliases. The XPRT system provides this capability with the Alias search query type. The following interaction diagram describes the sequence of operations that provide this capability.

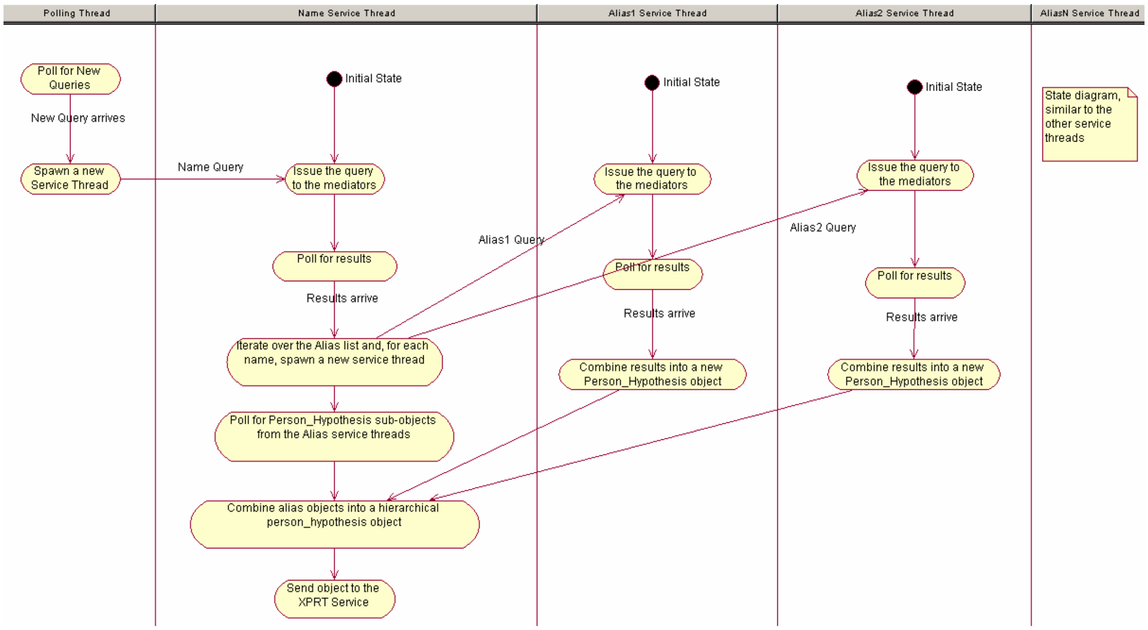

Polling Thread
Name Service Thread
Alias1 Service Thread
Alias2 Service Thread
AliasN Service Thread
Poll for New Queries
New Query arrives
Spawn a new Service Thread
Name Query
Initial State
Issue the query to the mediators
Poll for results
Results arrive
Iterate over the Alias list and, for each name, spawn a new service thread
Poll for Person_Hypothesis sub-objects from the Alias service threads
Combine alias objects into a hierarchical person_hypothesis object
Send object to the XPRT Service
Alias1 Query
Initial State
Issue the query to the mediators
Poll for results
Results arrive
Combine results into a new Person_Hypothesis object
Alias2 Query
Initial State
Issue the query to the mediators
Poll for results
Results arrive
Combine results into a new Person_Hypothesis object
State diagram, similar to the other service threads

Figure 10: Dynamic Diagram of the Name Alias Aggregator

1. The Polling thread of the AliasAggregator receives a new query. The polling thread spawns a new thread to process the results of the name search, called Name_Service thread. It then returns into polling for more queries.
2. The Name_Service thread first issues the name query to the mediators. It then polls for results.
3. When it receives a Person_Hypothesis result, it iterates over the Alias list of the object, and for each name in the list, spawns a new Alias_Service thread. It then goes into polling for results from the various Alias_Service threads.
4. Each Alias_Service thread issues a query for the alias name to the mediators and polls for results.
5. As it receives objects from the mediators, the Alias_Service thread assembles them into an appropriate instance of the Person_Hypothesis object and sends the object to the Name_Service thread. This object contains information for the respective alias of the original person.

The Name_Service thread adds the various Alias objects as children to the original Person_Hypothesis, and sends the full object to the XPRT Service.

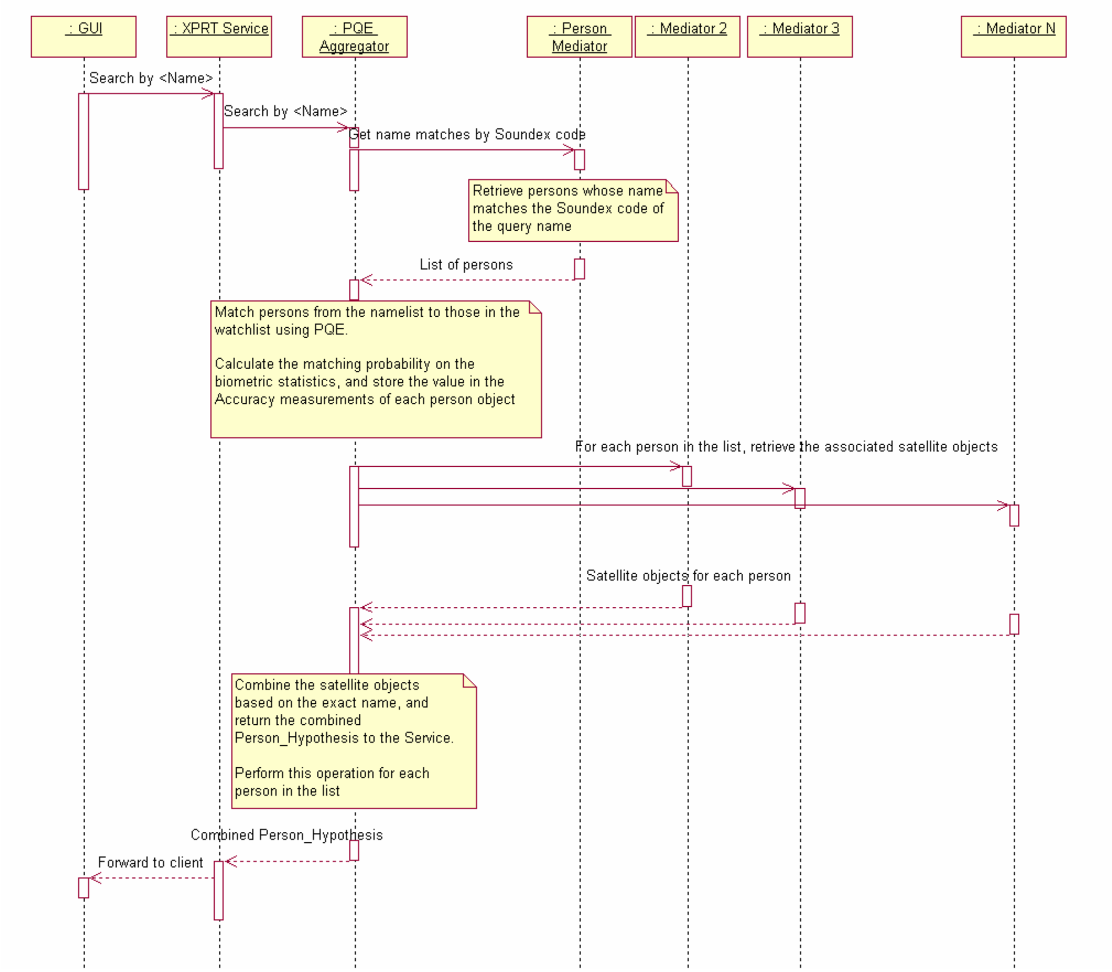
4.4.3 Probabilistic Match Search

The purpose of the Probabilistic Query Engine is to match individuals based on their biometric statistics. The name query retrieves results based on an exact name match, and as result, runs into the risk of missing information when the name is misspelled across the various sources. The PQE query will address this limitation by retrieving individuals whose name is phonetically similar to the query request. However, since the number of returned records might be fairly large, the PQE aggregator will order the results based on the distance of each suspect’s biometric stats to those of the requested individual.

Biometric stats are usually inexact measurements and quite often are expressed as ranges of continuous values, or sets of distinct values. For example, the values in the Weight biometric can be expressed as ‘[150 – 160], U’, meaning that the value in this attribute is uniformly distributed in the range of 150 to 160 lbs. Similarly, the values in the 'Hair Color' Biometric can be expressed as '{Brown, Dark Blond}, U', meaning that an exact characterization of hair color was not possible, therefore it is equally possible (uniformly distributed) that the hair color is brown, or dark blond. Because biometrics are probabilistic in nature, we will use the Probability Query Engine to match these types of attributes.

The Name attribute, on the other hand, is deterministic, but is often spelled slightly different on the various sources. For that reason we will use Soundex [22] as the matching algorithm to compare names. Soundex was first developed in the early 20th century by the U.S. Census to identify genealogical relationships between individuals with similar names. Soundex was designed to work with English type names only. Today, however, Soundex is used in a plethora of matching applications and it is applied on names of ethnicities other than English. Soundex is simple and has been proven to be quite accurate, as compared to other, newer and more complex name matching algorithms. Aside from its simplicity, Soundex is currently implemented in Oracle, SQLServer and MySQL, and is also available by open-source code in Java and other languages.

As described above, the probabilistic match is divided into two operations, identify candidate individuals based on the Soundex match of the name, and then rate the closeness of the candidates to the requested individual by comparing the biometric statistics. The diagram below illustrates the sequence of operations for the execution of the probabilistic match.


: GUI
: XPRT Service
: PQE Aggregator
: Person Mediator
: Mediator 2
: Mediator 3
: Mediator N
Search by <Name>
Search by <Name>
Get name matches by Soundex code
List of persons
For each person in the list, retrieve the associated satellite objects
Satellite objects for each person
Combined Person_Hypothesis
Forward to client

Retrieve persons whose name matches the Soundex code of the query name

Match persons from the namelist to those in the watchlist using PQE.

Calculate the matching probability on the biometric statistics, and store the value in the Accuracy measurements of each person object

Combine the satellite objects based on the exact name, and return the combined Person_Hypothesis to the Service.

Perform this operation for each person in the list

Figure 11: Probabilistic Match interaction diagram

The PQEAggregator will first publish the query to Person mediator, along with a parameter that indicates the use of the Soundex match on the name. The Person Mediator will then retrieve persons from the database whose name matches the Soundex code of the requested name. It will then assemble all the candidate names and place them in the SimilarNames satellite object of the Person_Hypothesis. Finally, it will send the Person_Hypothesis object to the PQEAggregator.

The Aggregator will compute the probabilistic match between the internal watch-list table and the list of candidates returned from the Person Mediator. The match will be applied over the four biometric attributes, Weight, Height, Hair-Color and Weight-Color. The result of the probabilistic match is stored in the Probability attribute of the PersonHypothesis object.

The matched individuals and their probability of match are then returned to the GUI for display.

4.4.4 Unstructured Data Search

We implemented the Unstructured Data Search capability using the Oracle Text technology [1]. This technology is summarized in the aforementioned paper as follows "Oracle's Text engine is a theme-based retrieval method implemented on top of traditional Boolean techniques. Theme-based retrieval enables querying for documents that are *about* a certain theme or concept. The set of themes from a document together define what a document is about. Themes are extracted from documents and queries by parsing them using an extensive knowledge-based lexicon together with an association catalog of concepts and relations. High precision is achieved by a disambiguation and ranking technique called *theme proving* whereby a knowledge base relation is verified in the lexical and semantic context of the text in a document."

Oracle maintains the lexicon and the association catalog. Our experience with the shipped knowledge-base resources (both lexicon and catalog) was that they needed to be extended in order to use them for the anti-terrorism use case. The knowledge-base resources, for example, did not have an association between several known terrorists and the term terrorism. It is not clear how users can extend these resources to add new terms and new association between terms. Despite these limitations, however, the Oracle text was particularly effective for exact keyword searches and also for theme based queries of well-known terms. For example, a query on Colin Powell returned several FBIS documents that contained references to the U.S. State Department. Moreover, these documents were ranked by the probability of the match to the original query requests, and the documents that have direct report to the Secretary of State were listed at the top of the list. Oracle text was also fairly quick retrieving documents for exact keyword search, but considerably slower for theme based queries.

The diagram below describes at a high level the interactions between components that implement the Unstructured Data search, and the subsequent paragraphs provide the description of each component's functionality.

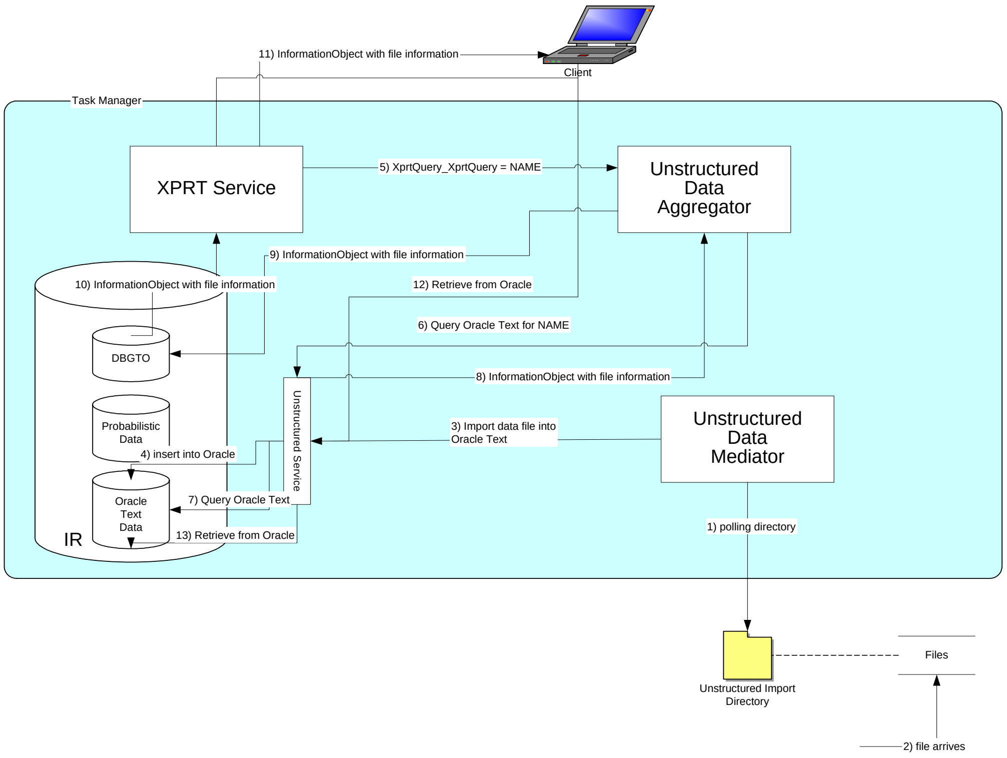

11) InformationObject with file information
Client
Task Manager
XPRT Service
5) XprtQuery_XprtQuery = NAME
Unstructured Data Aggregator
9) InformationObject with file information
10) InformationObject with file information
12) Retrieve from Oracle
6) Query Oracle Text for NAME
DBGTO
8) InformationObject with file information
Unstructured Service
Probabilistic Data
4) insert into Oracle
3) Import data file into Oracle Text
Unstructured Data Mediator
Oracle Text Data
7) Query Oracle Text
13) Retrieve from Oracle
IR
1) polling directory
Unstructured Import Directory
Files
2) file arrives

Figure 12: Unstructured Service Components

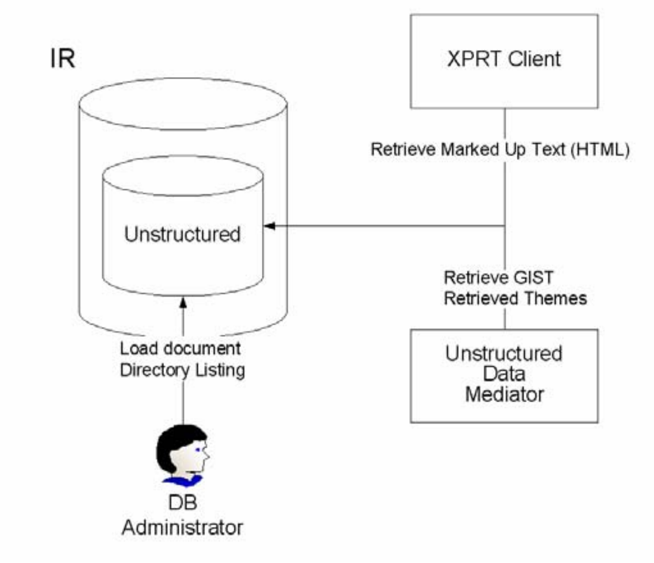
The Unstructured Data Aggregator receives a Name query from the XprtService and executes a text query against the Oracle Text partition of the IR via the Unstructured Service. From the results of the query, it constructs InformationObject objects and publishes them to the IR, indicating which files match the query criteria.

The Unstructured Data Mediator continually loads the files from the Unstructured Import Directory into the Oracle Text partition of the IR via the Unstructured Service. The Unstructured Import Directory should be configurable and must be accessible to the XPRT Service.

The Unstructured Service provides an abstraction for the Oracle Text partitions of the IR. It will provide interfaces to allow the ingestion, query, and retrieval of data. It provides the following functionality:

1. Load a file from a specified directory into the Unstructured Partition of the IR.
2. List the files available for loading into the Unstructured Partition of the IR.
3. Retrieve the GIST of a document
4. Retrieve the THEMES of a document
5. Retrieve the marked up version of a document in HTML

Currently the Unstructured Partition of the IR is implemented using Oracle Text. A high level diagram indicating interactions is presented below.



4.4.5 Integration with the Group Detection Algorithm (GDA)

For group detection we used the GDA algorithm from CMU. The Group Detection Algorithm uses maximum likelihood estimation to find groupings of entities from two sets of data. The demographic data set that contains information about individuals, i.e. title, affiliation, etc., and the link data set consists of a listing of links between individuals. GDA is agnostic to the type of the link. In a real operational environment, the demographic and link data will be spread across multiple external sources. In our prototype however, all the data reside in a single database, the Evidence DB. This database is used extensively by the EAGLE program community and contains association between individuals with many different link types. The simulated EDB data set contained only seven different link types. Even though we restricted our data sets to one database, extending our work to handle multiple external sources can be easily accomplished as part of a hardening process to a real operational system. This work will require the development of appropriate data mediators and aggregator processes that are cognizant of the target database schema.

This section describes the components that are used to implement the integration of GDA with XPRT. It does not describe the capabilities of the GDA application. For a description of GDA see [19]. The diagram below describes at a high level the interactions between the various XPRT components. The subsequent paragraphs provide the description of each component's functionality.

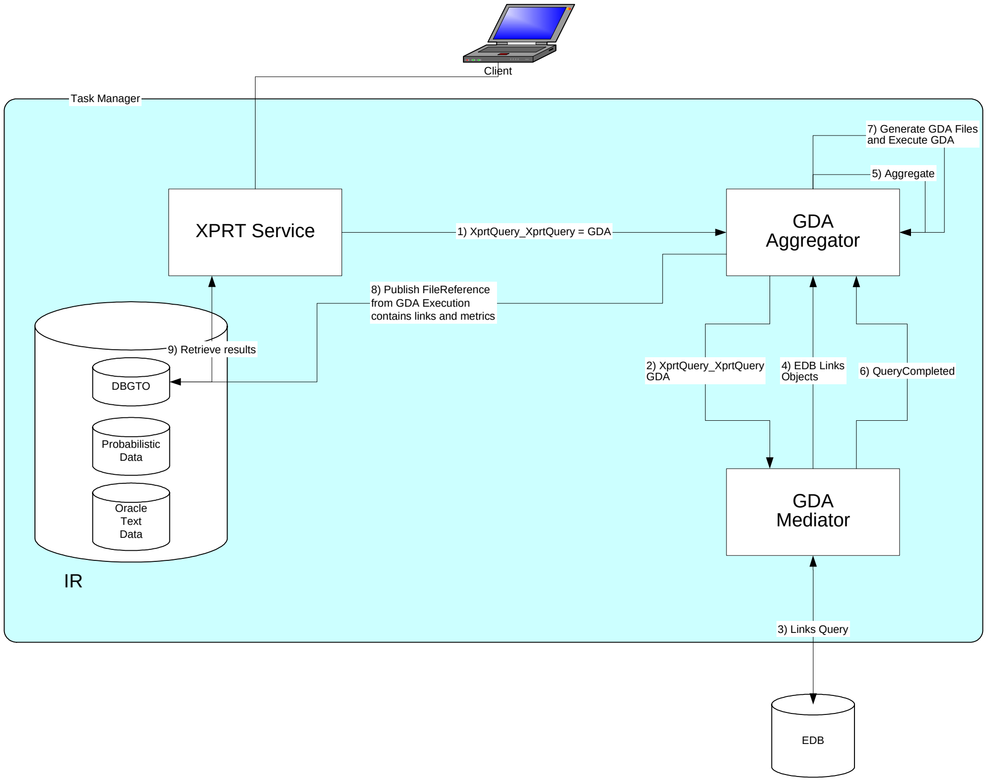


Figure 13: GDA Components

The GDA Aggregator receives a GDA query from the XPRTService and publishes the GDA query to the GDA Mediator. After the GDA Aggregator publishes the query to the GDA Mediator, it will poll its subscription channel, aggregating the results until it receives a QueryCompleted message. Once it has received the QueryCompleted message, it will create the links and demographics files required for GDA and invoke GDA. Once GDA has completed executing, the GDA Aggregator will publish a InformationObject ontology object to the XPRTService indicating where the GDA results exist.

When the GDA Mediator receives a GDA query from the GDA Aggregator, it generates a query to retrieve all or the link type specified in the query, depending on the parameters, and populates EDB ontology objects. It publishes the EDB objects back to the GDA Aggregator. When the GDA Mediator has completed its query, it will publish a *QueryCompleted* message object to indicate there are no more links.

5 – RESULTS

The purpose of our testing was to evaluate the behavior of the system under a typical workload given the processing capacity of the test hardware. The test hardware consisted of two server machines.

- XPRT server: A single-CPU windows server, 2.7 GHz Pentium-4 processor and 1 GB of RAM. This server executed all the component processes of the XPRT prototype. The component processes consist of one process for the XPRT service, 5 processes for aggregators, and 19 processes for the various data mediators.
- Database Server: Two Sun 280R systems with 2 CPUs each, 900 Mhz SparcIII processors with a total of 16 MB of eCach and 2 GB of RAM. The database is an Oracle 9.i Enterprise Edition server, highly optimized for fast data access.

The data set consisted of four database schemas representing four simulated relational databases and two web sources, Google news and BBC. For detailed information about the type of the data-sources, their statistics, and the mapping of data source to the mediators and the domain ontology please see Table 1 at page 8.

In a real operational environment the various Data Mediator components will be running on separate machines, preferably on the same machine as the associated database server. In addition, the various aggregators could be distributed across multiple components of a parallel hardware system. In our simulated environment, however, we run all the XPRT components on a single machine. We estimate that 10 to 20 concurrent queries will be representative of a typical workload for our simulation environment. We evaluated the behavior of the XPRT prototype based on this assumption.

We performed two sets of experiments. In the first set we compared the performance of the system as we increased the number of concurrent queries. In the second set we compared the performance improvement that the IR provides for cached results vs non-cached ones. The queries in the first experiment retrieve data solely from the external sources.

5.1 CONCURRENCY TEST

To test the system we incrementally submitted a new query search every 2 to 3 minutes for a total of 15 concurrent queries. At any given time the mix of the queries consisted of 50% name queries, 30% aliases and 20% biometrics. A typical query takes about 4 to 5 minutes to return all the data back from the various databases. Most of this time is spent in database operations rather than processing by the XPRT system. Since Oracle does not provide tools to measure its processing time and also delineate the processing load between the database and the client application, we used the number of open Oracle sessions as a surrogate metric for the database load. In other words, when the number of open Oracle connections increase, we assume that the database server incurs higher load.

During our experiments we measured the following four statistics: the time it takes to return the first set of results for a query, the number of open Oracle sessions, the number of the system threads, and the size of the virtual memory.

The results are summarized below:

Query Name	Time for First Data to Return (sec)	Oracle Sessions	System Threads	Memory Size
IDLE (GUI Connected to Service)	0	0	1058	2862536
Saif al Adel	20	4	1160	2999608
Anas al Liby	10	6	1157	3011088
Mustafa Ahmed Hassan Hamza	10	8	1160	3017944
Colin Powell	13	8	1170	3022396
Ella Cole	34	11	1171	3024024
Ed Veale	11	15	1173	3024128
Joe Cho	26	15	1177	3026852
Brian Murphy	21	8	1173	3032716
Davis Ramirez	16	12	1176	3033456
Amy Lee	11	12	1181	3036016
Jim Ray	21	9	1181	3038732
Ian Ford	14	11	1183	3038700
Judy Low	13	10	1188	3041212
Yu Smith	16	12	1190	3042248
Erik Lee	24	13	1194	3041232

The data are also displayed in a graphical form below:

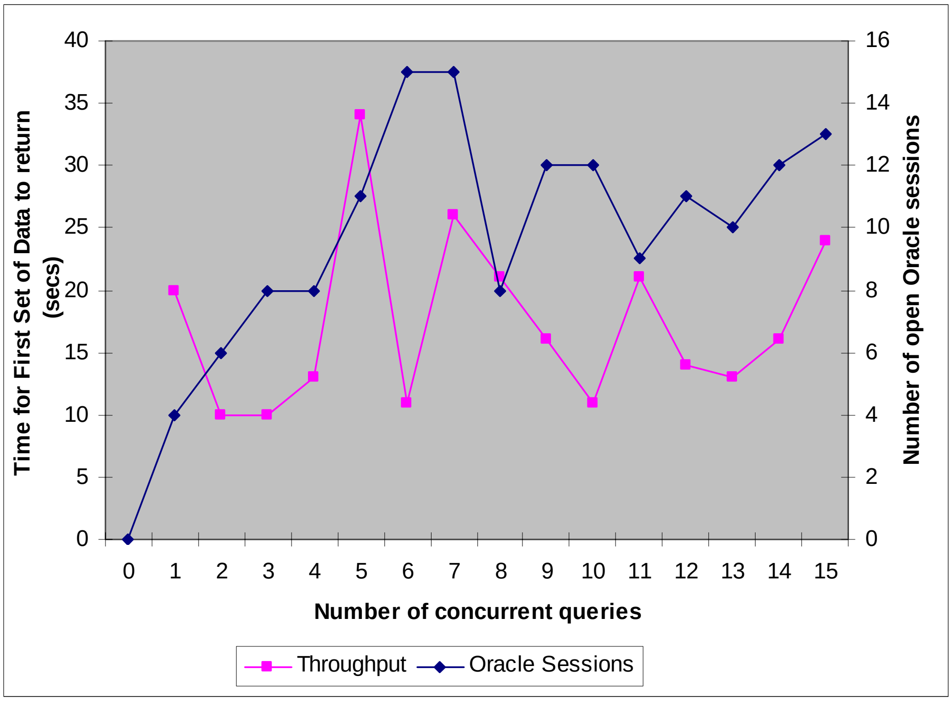


Figure 14: Throughput

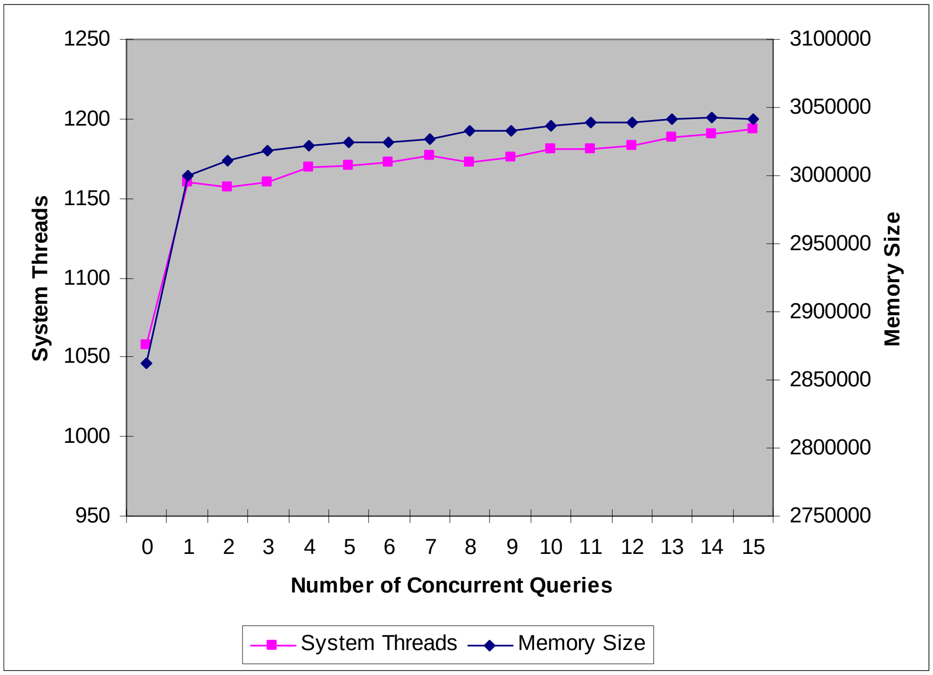


Figure 15: System Resources

Given these graphs, we can draw the following conclusions:

- The throughput of the system oscillates throughout the simulation. We did not apply any statistical measurements to compare the behavior of the throughput, but empirically we observed that the time it takes to retrieve new information does not increase as we submit new queries. This was a functional objective of XPRT and it was achieved via the use of the distributed processing architecture and the multi-threading implementation.
- The Oracle sessions reach a steady state after eight concurrent queries. The database processing increases almost linearly at the beginning of the simulation, and it reaches a steady state of 12 to 14 concurrent open sessions at the eighth concurrent query.
- The number of the system threads increases when we issue the first query, and it reaches a steady state after that. This behavior is due to the connection pooling implementation.
- The system memory increases, also, with the submission of the first query, and then it also reaches a steady state. We observe that, for each additional query, the memory size increases insignificantly to cause any memory problems. Towards the end of the simulation the memory size even decreased.

These experiments demonstrate that the system can handle a load of 10 to 15 concurrent queries without any significant performance degradations. In a typical operational environment with hardware systems of ten times higher processing capacity over our simulated environment, we expect that XPRT will be able to sustain a load of 100 to 150 concurrent users.

5.2 INTERNAL REPOSITORY TEST

In addition to the above tests, we also compared the performance improvements of cached queries vs. non-cached ones. This experiment attempts to quantify the benefits of the Internal Repository in terms of performance improvement. We structured our experiments as follows:

- We populated the Internal Repository with a fixed number of person hypothesis objects and all their associated sub-objects. We initially inserted information for 200 persons in the database, which resulted in approximately 700 associated transactions (i.e. hotel transactions, aliases, flight records, INS Visas, etc).

  The number of 200 people that we choose to start with is 0.1% of the total population size in our simulated universe (there are 204102 persons in all our external sources). We believe that 0.1% to 1% size of a total population is a reasonable size of data that can be accommodated by the XPRT cache in a real operational environment. In a real counter-terrorism system, there could be hundreds of millions of individuals, along with their associated transactions spread across several dozen of databases. In that environment, 1% of the total population will result in 1 to 10 million individuals stored in cache at any given time. We believe that this volume represents a realistic scenario.
- We issued three query types, Name, Alias and Biometric against three new individuals that have lots of transactions. These queries resulted in requests against the external databases. We measured the time it took to retrieve all the results from each one of the above queries
- We then issued the same three queries again. XPRT detected that these queries have already been executed, and in this case it retrieved the data directly from the Internal Repository. We measured again the time it took to get all the results back to the GUI.
- We then increased the size of the Internal Repository with an additional 200 persons, which now resulted in a total of 1800 associated transactions.
- We performed the same set of operations as before (run three new queries against the external sources and the IR) and we compared the results.
- The names of the people that were used to populate the IR were chosen from the Centaurus database, and as a result they contain a moderate number of associated transactions. On the average, each person has 5 associated transactions. This process ensures that the Person-Hypothesis objects cached in the Internal Repository contain several sub-objects.
- For the biometric query we chose a common name that returned back 350 potential matches.

The performance improvement of the IR vs the External Sources search is displayed in the following figure:

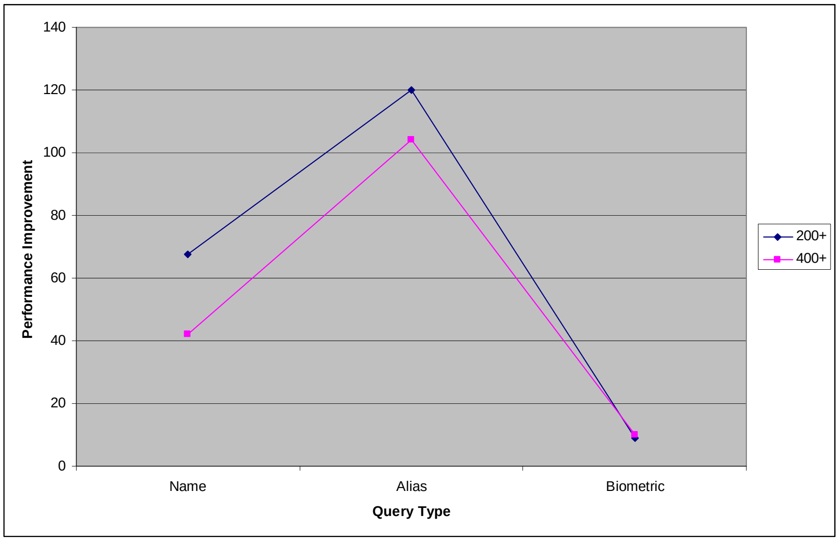


Figure 16: Internal Repository Performance Improvement

The above figure shows the performance improvement of IR with 200 and 400 individuals in the cache. The performance improvement is defined as the ratio of the time it takes to retrieve all the results back from the external sources over the time it takes to retrieve the same results from the Internal Repository.

We observe that the Internal Repository provides a performance improvement between one and two orders of magnitude. This is the results for the following three reasons. First, the size of the internal repository is considerably smaller than the size of the external sources. Secondly, the structure of the objects persisted in the IR match the ontology of the domain problem. Since we have already retrieved the sub-objects from their sources, and fused them in the structure specified by the domain ontology, when we access these objects from the IR we don't incur any additional transformation operations. As a result, the access time is faster. The third reason is that when we access data across multiple sources, we need to perform cross database joins on the XPRT server. Since the XPRT server is not optimized for database operations – database servers have been steadily improving their query optimizers and the relational algorithms for several decades now – we incur delays considerably longer than if the same set of data existed on one server and the retrieval operations were executed solely by the database.

We observe that the performance improvement of the Alias query type is larger than the name query. This is expected because the Alias query returns more data back to the users, therefore the retrieval operations take longer to execute than simply extracting the cached data from the IR. The result of this behavior is that the IR will perform better as the number of returned objects increases.

The improvement of the Biometric query is one order of magnitude, and it is the lowest of the other two. This result might be an artifact of the way we had structured our experiments. The names of the individuals that are retrieved from the biometric query, along with their biometric information, resides in a single data source, Centaurus. Consequently, when we perform the biometric query against the external sources, in effect we only access data from one such external source. As a result most of the database operations are localized in one schema, therefore we don't pay the extra cost of cross-database joins on the XPRT server.

Finally, we observe that the performance improvement slightly decreases as the size of the Internal Repository grows. This is expected, because the database server has more data to sift through in order to return the results of a query. We have not experimented with larger size IRs, but we don't expect the performance improvement to drop significantly.

6 – CONCLUSIONS

We developed a system that fuses information from multiple external sources, utilizes an Internal Repository for fast access and handles uncertainty. We evaluated our architecture with a domain ontology tailored for the Counter-Terrorism problem and a series of external sources, some populated with simulated data and others with information from open source. Finally, we designed the system to be a virtual data repository for various Link and Group Understanding tools, and we tested this capability with one such tool, the Group Detection Algorithm.

We used multi-threading and distributed processing to achieve high performance and scalability. Our experiments showed that this architecture was effective, and that the performance of the system does not decrease as the number of concurrent queries grows. Moreover, because of our tight memory management and thread pooling policies, we managed to host the whole XPRT system, with the exception of the database server, on a single windows machine. In a typical operational environment, we expect that the various XPRT component processes will be distributed over dozens of servers.

The Internal Repository provides a significant performance improvement over the external sources. Our Ingestion Service consists of an efficient Object to Relational mapping mechanism that maps domain objects to relational structures and efficiently persists and retrieves complex java objects to the database with a simple programmatic interface. Our ingestion service manages continuous streams of data and has the ability to rank information based on the temporal order of their persistence in the Internal Repository. This is particularly important for the GUI that needs to differentiate new versus old information and render it differently in the results panel.

We developed processes to retrieve data from the Evidence Database, mediate them to the GDA format and then run GDA against that data. The results of GDA were rendered by the XPRT GUI using our LinkView visualization tool. This effort demonstrates that we can integrate Link and Group Understanding tools and that XPRT can provide the common repository between external source, analysis tools and visualization applications.

The probability query engine was used to rank individuals based on the match of their biometric statistics. Biometrics is increasingly used by the various Law Enforcement and Intelligence Agencies to screen suspects. With the expansion of the U.S. Visit program, where non-US citizens visiting the US are required to provide their biometric data, we believe that algorithms that can operate on biometrics and manage uncertainty will become particularly important to the Intelligence Community.

Finally, we built a GUI in Java that facilitates the submission of queries and the visualization of the results. Our GUI has some unique characteristics, not readily available in other applications. The format of the results panel automatically adapts to the structure of the domain ontology as this ontology evolves. This means that the GUI does not require any code changes to display objects that have evolved from their original state. Evolution is defined in the strict terms of object oriented sub-classing and derivation. This capability is particularly important for enterprise systems where components, databases or applications, can slightly evolve their part of the ontology without coordination with the other components. The GUI automatically displays new information as it becomes available in the source databases, and the new information is rendered differently from the old data. This capability makes the GUI behave as a window into an ever changing source of data. Finally, the components of the GUI are fairly independent so that other applications can be integrated fairly easily. This makes the XPRT GUI behave as a dashboard application where new capabilities can easily be added via the use of plug-ins.

Our experiments and our results demonstrate that we addressed the majority of the Genisys needs and we proved that such a system will be beneficial to the Intelligence Agencies. As a result of this activity we have gained insight for future needs, particularly with respect to integrating with Link and Group Understanding (LGU) tools, and we plan to apply this experience in other information management projects.

7 – REFERENCES

1. Kavi Mahesh, Jacquelynn Kud, Paul Dixon, 'Oracle at Trec8: A lexical Approach', http://otn.oracle.com/products/text/htdocs/imt_trec8pap.htm, *Proceedings of The Eighth Text REtrieval Conference (TREC-8)*.
2. David Grossman, Steven Beitzel, Eric Jensen, Ophir Frieder, 'IIT Intranet Mediator: Bringing Data Together on a Corporate Intranet,' IT Pro, January-February 2002
3. David Grossman, Ophir Frieder, David Holmes, David Roberts, 'Integrating Structured Data and Text: A relational approach,' *Journal of the American Society of Information Science*, 48(2), February 1997
4. O. Frieder, D. Grossman, and A. Chowdhury, "On Scalable Information Retrieval Systems," Keynote Paper, *IEEE Second International Symposium on Network Computing and Applications*, Cambridge, Massachusetts, April 2003.
5. Steve Silberman, 'The Quest for Meaning,' Wired Magazine, February 2000
6. Clara Yu, John Cuadrado, Maciej Ceglowski, J Scott Payne, 'Patterns in Unstructured Data: Discovery, Aggregation, and Visualization,' http://javelina.cet.middlebury.edu/lsa/out/lsa_biblio.htm
7. Metamatrix, 'Enterprise Information Integration,' www.metamatrix.com
8. Metamatrix, 'Leveraging Enterprise Data Assets,' www.metamatrix.com
9. Metamatrix, 'Model Driven Information Integration,' www.metamatrix.com
10. Laks V.S. Lakshmanan, Nicola Leone, Robert Ross, V.S. Subrahmanian, 'ProbView: 'A flexible Probabilistic Database System,' ACM Transactions on Database Systems, September, 1997
11. Thomas Eiter, James J. Lu, Thomas Lukasiewicz, V.S. Subrahmanian, 'Probabilistic Object Bases,' ACM Transactions on Database Systems (TODS), Vol. 26, 264-312, September 2001.
12. Dyer, D., "Genisys", DARPA IAO PAD ID Number 020115, Program Number QGSYE, Version 9, October 2002.
13. ALPHATECH, Inc, 'eXtensible Probabilistic Repository Technology (XPRT),' BAA 02-08 Response, 21 April, 2002.
14. B. Krikeles, "eXtensible Distributed Architecture Query Guide", January 2002, ALPHATECH Report.
15. B. Krikeles, A. Lusignan, E. Starczewski, "eXtensible Distributed Architecture (XDA): A Framework For Distributed Data Fusion", *2001 MSS National Symposium on Sensor and Data Fusion*, June 2001.
16. B. Krikeles, A. Lusignan, E. Starczewski, "Adaptive, Distributed Fusion for Battlespace Awareness", *2000 MSS National Symposium on Sensor and Data Fusion*, June 2000.
17. T.J. Rogers, R. Ross, and V.S. Subrahmanian, "IMPACT: A system for building agent applications," *Journal of Intelligent Information Systems*, Vol. 14, 95-113, 2000. (see http://www.cs.umd.edu/projects/impact/ for information about project IMPACT).
18. ALPHATECH, Inc, 'XPRT User's Manual,' July 2004.

19. J. Kubica, A. Moore, and J. Schneider. ‘Tractable group detection on large link data sets’, In *CMU Tech. Report 03-32, 2003,* http://www-2.cs.cmu.edu/~jkubica/papers/kgroups_ICDM.pdf

20. B. Ludäscher, A. Gupta, M.E. Martone, “Knowledge-Based Mediation and XML-Based Information Ingtegration,” *17th Int’l Conference on Data Engineering, IEEE CS Press, Los Alamitos, California, 2001*.

21. Barry Silk, Byron Bergert, ‘EAGLE Evidence Database (EDB) Description,’ EAGLE portal, http://www.eagle-link.org, Global Infotek, Inc. (GITI) documents.

22. The Soundex Indexing System, http://www.archives.gov/research_room/genealogy/census/soundex.html

8 – APPENDIX A: XPRT CONOPS

Scene	Data	Technology
Scene 1 Jane is one of the Terrorist Threat Integration Center (TTIC) analysts responsible for monitoring the al-Qaeda organization. As part of her duties, she maintains a watch list of 10 al-Qaedea individuals. This includes: • Maintaining basic bio information and monitoring the movements, communications, and activities of each individual on her watch list • Providing individual-level “indications and warning” for individuals on her watch list Jane sits down this morning (Tues, 6 July 2004) and looks at an overnight XPRT search that she sets to run once a week. This search looks for any movement or communications by her targets/aliases. The search is run on Monday, 5 July 2004 The results turn up 2 flights, 3 coms (one via unstructured data in a NSA report), and 2 HUMINT sightings for a total of 5 target “hits.”	Results of query shows flight manifests on two names (one under Alias), and communications on one of those names from structured data. 3 other names will have information that is found in the unstructured data tab, including 2 HUMINT sightings and 1 NSA transcript of a phone call.	XPRT • Data mediation • Pedigree User Interface • Log in window • Query interface • Results. **Benefits** • Coordinated view across multiple data sources • Direct links back to original database sources for drill down.

Scene	Data	Technology
Scene 2 Looking through the returned data, Jane sees that one of the people on her watch list (Mustafa Hamza but using alias Mohamed Gamal El-Sayed) was listed on a flight manifest for a flight from Heathrow to JFK on June 30th. INS records show that he arrived at 1715 that evening. He is staying at the “ X” Hotel in NY. Another person from the watch list, Saif al Adel, arrived in NY on 28 June and is staying 6 blocks away at the “X+6” Hotel. Jane looks under the communications tab to see if Hamza/Sayed or Saif has been contacting anyone from NY. The results indicate that Saif called two people in NJ on 7/1 by cell phone. Both calls went to previously unknown numbers in NJ. Jane will ask the NSA to provide transcripts and the FBI to examine the numbers and try and get more info on the persons called.	Centaurus hotel info, flight manifests, and INS records. EELD communications data, including cell phone calls…	XPRT ● Data mediation ● Query language User Interface ● Query interface and templates/sub-templates. **Benefits** ● Coordinated view across multiple data sources.

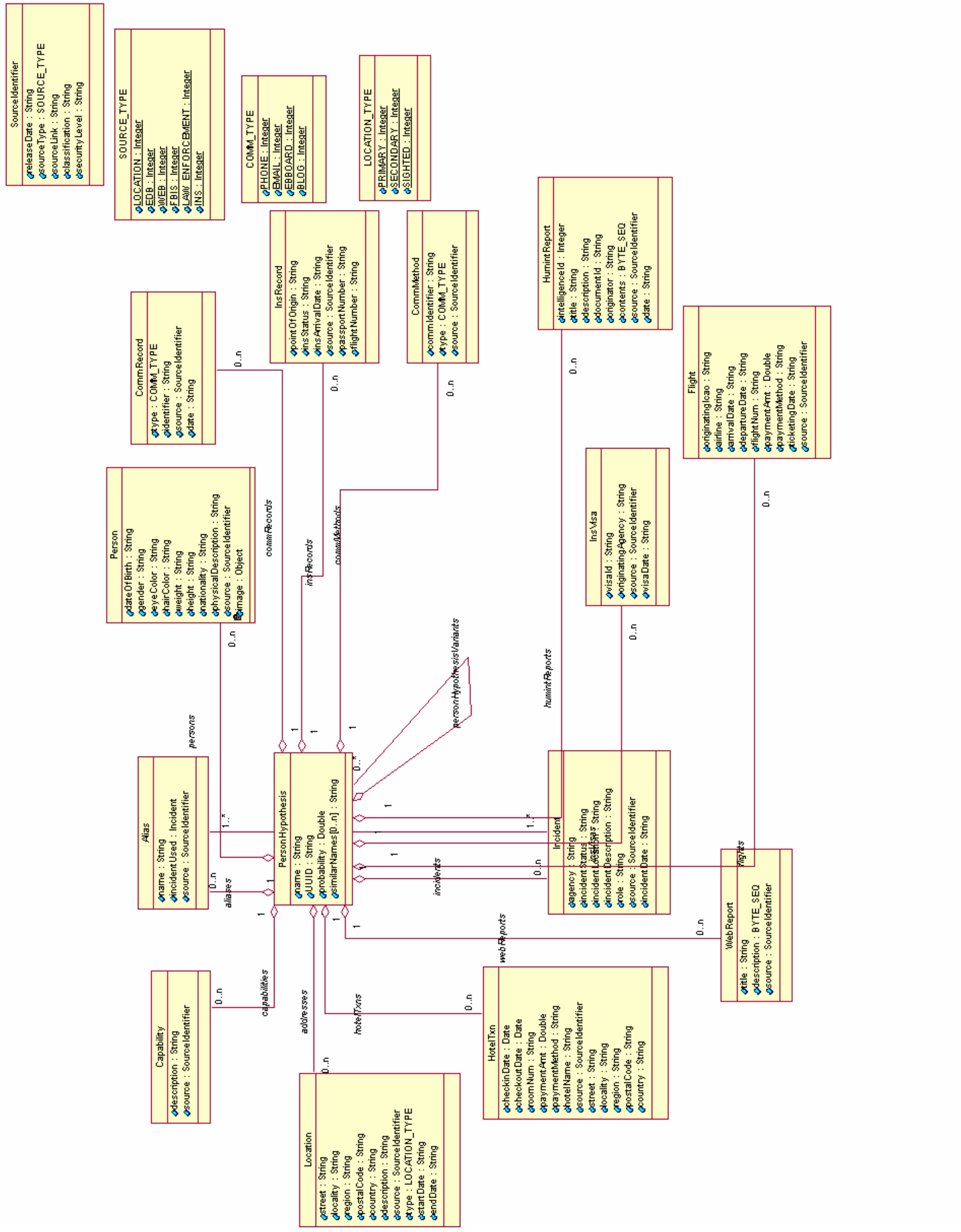
Scene	Data	Technology
Scene 3 Before continuing her analysis, Jane decides to make another query to see if Mustafa Hamza and Saif al Adel have traveled together before. These could include: ● In the past 3 years, have they ever taken the same flight or w/in one week of each other; ● Have they ever traveled within 50 miles and 2 weeks of each other; ● Have they ever used the same credit card. We could mention that: Jane could also decide later to set up a query that would see if there are any other people who routinely show up in the same hotel or on flights as Mustafa Hamza when he travels…possibly to add them to her watch list.	Flight manifest, EELD data on phone calls ?Any transcripts of past phone calls… makes sense that they would exisit The Analyst actual watch list with some information under all of the "tabs"	XPRT ● Data mediation ● Query language User Interface ● Query interface and templates/sub-templates. **Benefits** ● Coordinated view across multiple data sources. ● Easily define new standing queries based on templates.

Scene	Data	Technology
Scene 4 Jane realizes that there are other organizations that will want to know about Hamza/Sayed's and Saif's visit and their contacts. She also knows she does not have access to all of the databases that may already hold relevant information about her target's activities in New York. In particular, Jane knows that there is an FBI database that she does not have routine access to that lists suspected al Qaeda members in the greater New York area. Jane wants to see if the FBI knows anything about the activities of Saif's phone contacts, Abdul al Mugrin or Muhamar Oled. Jane is also aware that there may be other databases that she may not have access to within the Department of Homeland Security, etc. She tells her counterparts at those organizations about her findings and asks them to search their databases for additional information.	Made up reports that FBI analyst finds from his database that Jane does not have access too.	XPRT • Security **Benefits** • XPRT's data mediation provides security to only allow a user to see data he/she has proper access too.

Scene	Data	Technology
Scene 5 After generalizing and saving the query and the results, Jane returns to the Watch List Window and reviews the data. Based on the results of the query, Jane decides to initiate a broader analysis to see if there is information on other people with similar names to Hazma and with approximately the same biometric data	The watch-list that Jane maintains, contains biometric information for all her suspects. In particular, Jane maintains height, weight, hair color and eye color. These biometrics are not exact, therefore instead of maintaining one value for each attribute, Jane maintains a range of values. For example, for Mustafa Hazma, the values of the biometric attributes are: **Height**: 5’7” – 5’10” **Weight**: 150 – 170 **Eye Color**: brown, green-brown **Hair Color**: black For the purposes of the demo the distribution of the values is uniform In addition to the watch-list, biometrics are also kept in the external databases. Some of the biometrics are single values, while some other are ranges of discrete values. All the biometrics, if available, are described by a uniform distribution.	XPRT must first search the databases and return transactions for individuals whose names have the same Soundex code as Hazma. Presumably this result set is larger than an exact name match. Then, XPRT should be able to compare the biometrics of Hamza’s record to the ones returned by the Soundex name match, and rank the results by the probability of the match. XPRT must be capable of handling continuous, discrete and null biometric values.

Scene	Data	Technology
Scene 6 Jane is now going to set up a data collection activity in order to generate enough data to feed to one of the application programs for further analysis. The target application is the Group Detection Algorithm (GDA), which takes as input co-occurrence data and demographic data. Jane will use the two targets that showed up in the same area, and their hotels as starting points. She can ask XPRT to collect the names of all the people who stayed in the same area, by zip-code, during a time window before/after the two people from the watch list, then look for the hotel stays of those people, then look for who else stayed at each hotel when they did, and so on. XPRT can also seek out demographic data that might be relevant (e.g., nationality, age, gender, etc.). Since Jane wants the data to be fed to GDA, XPRT will automatically prepare the data in a format that GDA can use.	Unstructured data: • HUMINT- and FBIS-enhanced EDB Structured data: • Centaurus • Watch lists	XPRT • Data mediation • Query language User Interface • Query interface **Benefits** • Easily query to generate a data set that can be fed to an external application. • XPRT's data mediation and search capabilities enhance the analyst's ability to use application tools by making it easier to get appropriate data into the format the tools require.

9 – APPENDIX B: PERSON HYPOTHESIS DOMAIN ONTOLOGY



10 – APPENDIX C: REQUIREMENTS

XPRT Service
The XPRT Service will encode incoming queries into GTOs for transfer to the Aggregators via its publish channels.
The XPRT Service will not require the client applications to “know” where the information physically resides
The XPRT Service will support read-only queries
The XPRT Service will operate with a predefined domain object model
In response to a query, the XPRT Service will return to the client a complete object in GTO format containing aggregated information from multiple sources.
The XPRT Service will notify clients when new objects become available.
The XPRT Service will support multiple concurrent client connections and multiple concurrent queries
The XPRT Service will allow clients to connect to the system at any time without disrupting system operation.
The XPRT Service will allow clients to disconnect from the system at any time without disrupting system operation.
The XPRT Service will publish status messages to file.
The interface will be implemented in CORBA using JacOrb
The XPRT Service will be based on XDA 4.1A
The XPRT Service will provide a well defined Interface
The XPRT Service will be a taskable XDA component
XPRT will assign a Universal Unique Identifier (UUID) to each of the subscriptions from the client.
The UUID Service must generate a UUID per run. Only one instance of the UUID Service will be running per XPRT system. The UUID Service must maintain a CORBA IDL interface allowing other programs to retrieve a new UUID.
The XPRT Service will provide the UUID as part of the query specification.
The XPRT Service should handle multiple subscriptions issued by a client. Each query will be treated with equal priority.
The XPRT Service must maintain a CORBA IDL interface declaring client interactions with the service (ex. openConnection(), createSubscribeChannel()).

The XPRT Service will be made agnostic to the type of ontology.
The XPRT Service must maintain a list of unique running queries. When a client subscribes to the XPRT Service, the XPRT Service must check to see if the query in the subscription message is already running.
The XPRT Service clients must be allowed to retrieve a list of running queries.
The XPRT Service clients must be allowed to attach to a specific running query; that is, the client can receive data resulting from a query issued by some other client.
The XPRT Service clients must be allowed to detach from a specific running query; that is, the client no longer receives data resulting from a query issued by it or some other client.
The XPRT Service closeChannel method will remove a specific subscription from the service.
When there are no more subscriptions for a particular standing query, the XPRT Service will terminate that standing query.
The XPRT Service will use the XDA DB_GTO interface to implement an Internal Repository (IR). The XPRT Service should retrieve data from the IR on existing queries, and it must also be able to pass new queries to the aggregator(s).
The XPRT Service must be able to publish security credentials issued by the clients.

GUI
Provide an easy to use graphical user interface that an end-user can use to query the XPRT system.
Be able to submit multiple queries concurrently.
Provide visual cues that indicate the status of running queries, i.e. query submitted, new data available, etc.
The GUI should operate asynchronously; it will return control to the user when it is waiting for the results of a query
The user should receive notification that their query was received successfully and that processing has begun. This notification should be apparent without being obtrusive.
Users should be notified when new information is available
Users should be able to retrieve new information at any time. Results will arrive asynchronously and will be highlighted differently from results that have already been viewed.
The results for a particular query will be presented to the user in a manner that explicitly indicates that the initiating query. Each internal frame should contain some information about the query, such as when the original query was issues and when the last update occurred.
The user should be able to easily navigate result sets from different queries.
The results set for a single query should be represented as a traversable display. Results will be represented as a set of tables on a panel – where each tab contains only a particular object type. On top of this tab, the results set will be presented as a sortable table, similar to table in Excel. Each object in a result set will be presented as a row in this table and new rows will be added dynamically when new data are retrieve.
Be able to save the query results into an Excel file. Each tab of the result panel will be stored as a separate worksheet in the Excel file.
New data should be added at the bottom of the result tables.
Hyperlinks will be active.
Standing queries will be displayed in the Standing Query Panel for ease of navigation.
Users must connect to the service with their credentials prior to issuing a query.
The results will be maintained in the Results Panel even after the connection to the service has been terminated.
Users should be able to view a list of the standing queries that are managed by the XPRT service.
Users should be able to attach to a standing query, depending on their authorization level, and view the results.

The GUI will support five types of queries, Name Search, Alias Search, Theme Search, Biometric Search and Group Detection.
The GUI must guide the user to formulate a Biometric Search query. This includes the entry of the biometric values with uncertainty.
The GUI must guide the user to interpret the results of a probabilistic query search. This includes the ordering of the results based on the probabilistic match.
The performance of a Biometric match must be comparable to the Exact-Name and Name-Alias queries. The performance objective must be met even when the potential number of returned individuals can reach hundreds of entries.
Biometric Match operations are batch oriented, not dynamic. New information – persons, updated biometrics, etc – in the external sources will be processed only after the Internal Repository is cleared and the client issues a new query request.
The GUI must provide a wizard to guide the user in selecting the records for input to the Group Detection Algorithm (GDA).
The output from GDA must be visualized as a graph with nodes representing individuals and edges represent ting inferred associations.
Indicate when a GDA query has completed.

Probabilistic Match
Users should be able to issue probabilistic match queries.
Probabilistic queries will match person names based on the Soundex code.
Probabilistic queries will match individuals based on four biometric statistics, height, weight, hair color, eye color
The ‘Height’ and ‘Weight’ attributes will contain ranges of continuous values uniformly distributed over the specified range. The ‘Eye color’ and ‘Hair color’ will contain discrete values uniformly distributed over the specified set.
PQE must support match against continuous probability values
PQE must support match against discrete probability values
The GUI must guide the user to formulate a probabilistic query. This includes the entry of the probabilistic values in the biometric statistics.
The GUI must guide the user to interpret the results of a probabilistic query. This includes the ordering of the resulting suspects by the probability match
The performance of a probabilistic match must be comparable to the Exact-Name and Name-Alias queries. The performance objective must be met even when the potential number of returned individuals can reach hundreds of entries.
PQE operations can only be executed by the aggregator against the Internal Repository. It cannot be assumed that PQE is available at the source databases
PQE operations are batch oriented, not dynamic. New information – persons, updated biometrics, etc – in the external sources will be processed only after the Internal Repository is cleared and the client issues a new query request
XPRT should be able to support multiple clients issuing PQE queries. This requires that XPRT forwards the proper list of matching individuals to the respective client.

Unstructured Data Search
In response to a search, the XPRT GUI will list the FBIS documents that contain pertinent information to the search token
The results will be shown as links to documents.
The results should be sorted by order of relevance. The relevance score is calculated by Oracle text.
The results should also contain a short description of the document with the relevant words highlighted
When the user opens a document, the query terms should be highlighted
The highlighted terms could be either the search term or the search theme. The search theme is calculated by Oracle text.
The Unstructured Mediators must be able to access the source file system directly and detect when new documents are available

Group Detection
The GUI must provide a wizard to start the GDA application
The GDA wizard must list the available links from the database, and allow the user to select the ones that GDA will use to calculate the groups. A list of the available links will be provided in the corresponding URD document.
The XPRT engine must be able to read the data from the EDB database and format the information in a manner that GDA can consume
GDA will operate against all the data from EDB
The output from GDA must be available to the XPRT/LinkView GUI.
The XPRT GUI must provide visual cues that GDA is running.
The XPRT GUI must indicate to the user when GDA has completed.
The user must be able to name the output file that holds the results of the GDA run.
The location of the GDA output file needs to be accessible by the server that runs GDA and the LinkView GUI
GDA operations are batch oriented, not dynamic. New information – persons, links, etc – in the external sources will be processed only after the Internal Repository is cleared and the client issues a new query request.