International Journal of Cooperative Information Systems

ACTIVE INFORMATION GATHERING IN INFOSLEUTH™

MARIAN NODINE, JERRY FOWLER, TOMASZ KSIEZYK,
BRAD PERRY, MALCOLM TAYLOR and AMY UNRUH
MCC
3500 West Balcones Center Drive, Austin, TX 78759 USA

Abstract

InfoSleuth is an agent-based system that can be configured to perform many different information management activities in a distributed environment. InfoSleuth™ agents provide a number of complex query services that require resolving ontology-based queries over dynamically changing, distributed, heterogeneous resources. These include distributed query processing, location-independent single-resource updates, event and information monitoring, statistical or inferential data analysis, and trend discovery in complex event streams. It has been used in numerous applications, including the Environmental Data Exchange Network and the Competitive Intelligence System.

Keywords: Multi-agent systems, agent-based systems, information agents, heterogeneous data, query processing, information subscription.

1. Introduction

In the past 15-20 years, numerous products and prototypes have regularly appeared to provide uniform access to heterogeneous data sources. As a result, that access to heterogeneous sources is now taken as a "given" by customers. Current MCC studies indicate that, given the availability of products that achieve heterogeneous data access, new needs emerge for solutions to the following issues:

- Dealing with information at different levels of abstraction and in varying media forms.
- Fusing overlapping information from multiple sources into integrated wholes.
- Monitoring and reacting to changes, or patterns of changes, occurring across the networked information sources.
- Adapting to a changing environment with respect to data availability and domain coverage.

In other words, it is the active and integrated exploitation of information from these sources at appropriate levels of abstraction that is of real concern to applications of online information networks.

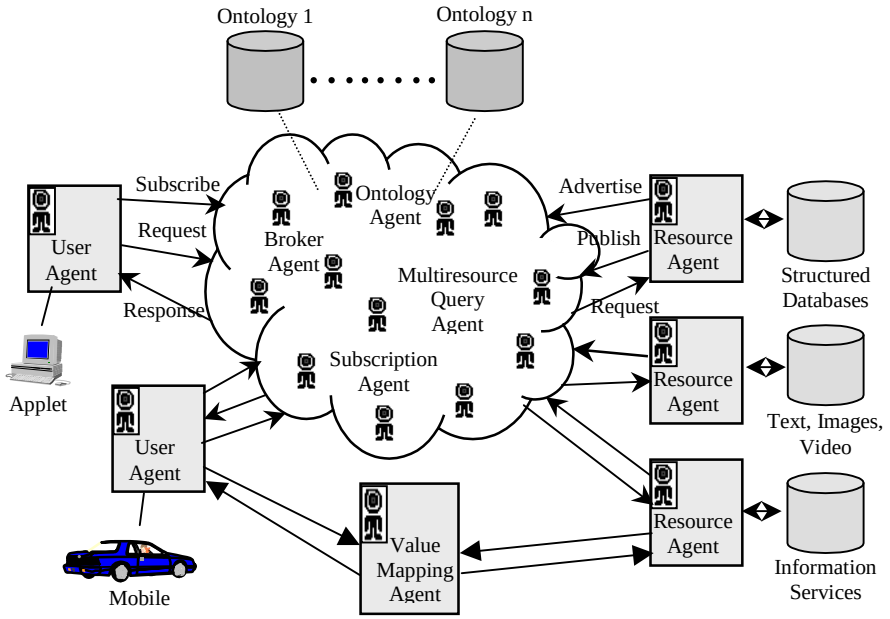

Fig. 1. Dynamic and broker-based agent architecture.

This paper provides an overview of the InfoSleuth project and its management of active information gathering and analysis tasks across heterogeneous information sources. InfoSleuth is an agent-based system that embodies a loosely coupled combination of technologies from information access, information integration, and information analysis disciplines. An agent system is a dynamic set of loosely interoperating (though cooperating), active processes distributed across a network or internet. Each agent in an information-gathering system is a specialist in a particular task in the information enterprise. Agents perform focused tasks for collecting data to create information at higher levels of abstraction or for collecting requests into generalizations of closely overlapping needs. Multiple agents interact and cooperate to solve complex information analysis tasks across multiple levels of abstraction. Typically an agent system uses some form of facilitation or market bidding to link up agents requesting services with agents providing services, and this capability is used to dynamically identify and compose goal-driven agent work groups.

We use the term "InfoSleuth" to refer both to an agent architecture for distributed information gathering and analysis and a deployed, advanced prototype implementation of that architecture. Figure 1 depicts the general model of the applications. An application domain is described by a set of *ontologies* that describe "domain objects, events and activities". The agent system for the application consists of a network of agents capable of performing information routing, analysis, extraction, and integration. A network of external information sources contains data

resources, at varying levels of abstraction, that provide partial evidence or facts for elements in the ontology. Resource agents wrap information sources, extract their content, mapping it to one or more domain ontologies, and monitoring their information. Users interact with the system by engaging in a session of ontology-based requests with a user agent. Core agents are used "off-the-shelf". The agent system dynamically identifies an agent interaction pattern appropriate to each ontological request for higher-level (i.e., integrated and derived) information artifacts, and uses this to decide which agents interact and exchange information to best satisfy that request.

This implementation is extended from that presented originally[1], in the following ways: we currently use OKBC (not KIF) for communication about ontologies. We have a wider variety of information resources, including text, images, as well as object-oriented databases and file-system based data. New agents exist for multi-resource query decomposition and recomposition, concept derivation, value mapping, complex event monitoring, enforcement of business rules and deviation detection.

The remainder of this paper provides an overview of various InfoSleuth applications and the agents and agent interaction patterns that make these applications possible.

2. Agent Organization

Figure 2 shows the basic classification of the agent functionalities. There are five categories of agents, represented in the figure by the four layers and one vertical box. Requests typically flow down from the user layer towards the resource layer, while data usually flow in the opposite direction. The arrows indicate that flow of requests and responses can flow between the agents in the layers at either end of the arrows. Depending on the nature of the request, different agent interaction patterns are possible, as discussed in Sections 8 and 9. Here we give a brief overview of the roles of the various agents.

User agents act on behalf of users to formulate their requests into a form understandable by the agents themselves, and transform results into a form accessible to the user.

Resource agents wrap and activate databases and other repositories of information. We consider any source of information a resource. Currently, we have implemented several different types of resource agents that can access different types of information. These include JDBC, text, flat files, and images. We also have service resource agents that start up offline data gathering and analysis services, such as web crawlers and document classifiers.

The remaining agents serve as the glue of the system, gathering information needed to process the users' requests, and synthesizing, filtering and abstracting that information into the level of abstraction that the user requires. These agents are classified as service agents, query and analysis agents, and planning and temporal

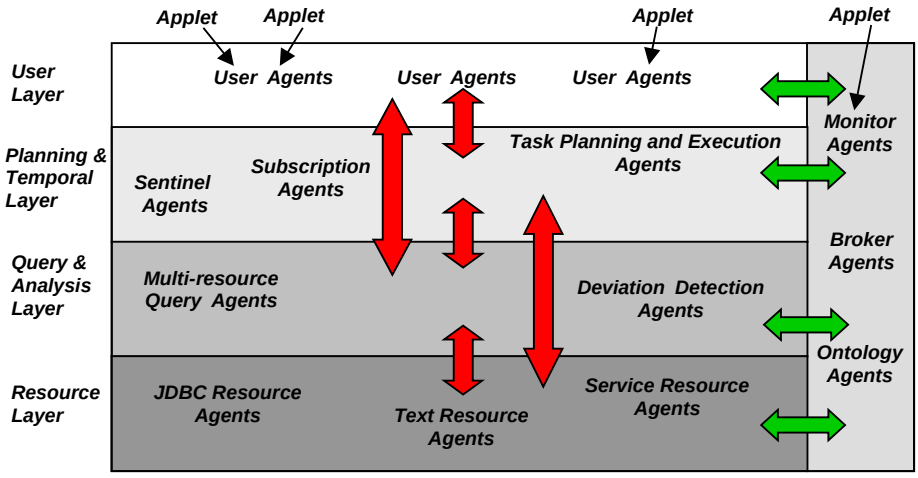

Fig. 2. Layers of agents.

agents. Service agents, represented in the vertical box along the side of the figure, provide internal information to the operation of the agent system. Query and analysis agents, represented in the layer above the resource agents in the figure, fuse and/or analyze information from one or more resources into single (one-time) results. Planning and temporal agents, represented in the layer below the user agents in the figure, guide the request through some processing which may take place over a period of time, such as a long-term plan, a workflow, or the detection of complex events.

Service agents include broker agents, ontology agents, and monitor agents. **Broker agents** collectively maintain a knowledge base of the information the agents advertise about themselves. Brokers use this knowledge to match requested services with agents. Technically, the brokers collaborate to implement both syntactic and semantic matchmaking. When an agent comes on-line, it advertises itself to a broker and thus makes itself available for use. When an agent goes off-line, the broker removes its advertisement from the knowledge base. **Ontology agents** collectively maintain a knowledge base of the different ontologies used for specifying requests, and return ontology information as requested. **Monitor agents** monitor the operation of the system.

Agents that do one-time query processing and/or data analysis include multi-resource query agents, deviation detection agents, and other data mining agents. **Multi-resource query agents** process complex queries that span multiple heterogeneous resources, specified in terms of some domain ontology. They may or may not allow the query to include logically-derived concepts as well as functions over slots in the ontology. **Deviation detection agents** monitor streams of data for instances that are beyond some threshold, where the threshold may be fixed or may be learned over time. Deviations themselves form an event stream for other

agents to subscribe to.

Agents that do planning or processing over time include subscription agents, task planning and execution agents and sentinel agents. **Subscription agents** monitor how a set of information (specified as an SQL query) changes over time. **Task planning and execution agents** plan how users' requests should be processed within the agent system, including how results should be cached. They may be specialized to particular domains, and support task plans tailored to their own domains. **Sentinel agents** monitor the information and event streams for complex events. A complex event is specified as a pattern of component events, which in turn may be other events such as changes in the information over time, triggers detected within individual resources, or deviations as detected by deviation detection agents.

Lastly, some domains require value mapping among equivalent representations of the same piece of information. User agents and resource agents may use specialized **value mapping agents** to assist them in this process.

3. Example Application Areas

InfoSleuth has been deployed in several applications. In this section, we discuss domain-specific features of two of those applications, Environmental Data Exchange and Competitive Intelligence.

3.1. Environmental information systems: the EDEN project

The Environmental Data Exchange Network (EDEN) is a collaborative effort of three United States Government agencies, the Environmental Protection Agency (EPA), the Department of Defense (DOD), and the Department of Energy (DOE), with the European Environment Agency (EEA). InfoSleuth is being used as a significant component of the EDEN project. The EDEN pilot demonstration enables integrated access via web browser to environmental information resources provided by offices of these agencies located in Idaho, Georgia, Maryland, Tennessee, Texas, and Virginia, with development of resources in Europe anticipated in the near future (Table 1 gives more detail). An auxiliary demonstration prepared for the Association of State and Territorial Solid Waste Management Organizations includes access to databases from the environmental agencies of several individual states.

The agencies involved in the EDEN project find the acquisition, use, and dissemination of environmental information to be of increasing strategic importance. Furthermore, congressional mandates have encouraged increased inter-agency cooperation in sharing data regarding environmental remediation efforts. InfoSleuth provides adaptability that may enhance the participants' ability to address changing business requirements: The domain ontology of an application will change with time, and data sources may come, go, and evolve.

Presented with an environment such as EDEN, with numerous legacy databases each with differing schema and often with different database management software, InfoSleuth provides a natural way of integrating data from the various sources by

Table 1. EDEN data sources

Database	Platform	Organization	Location	Content
CERCLIS	Oracle	EPA	Crystal City, VA	Superfund site profiles
HazDat	Sybase	EPA/CDC	Atlanta, GA	Toxicology information
ITT	MS-Access	EPA	Austin, TX	Remediation technology
EDR	Oracle	EPA	Austin, TX	Environmental Data Registry
ERPIMS	Oracle	DOD	Brooks AFB, TX	Air Force site profiles
IRDMIS	Oracle	DOD	Aberdeen, MD	Army site profiles
ERIP	Oracle	DOE	Idaho Falls, ID	DOE site profiles
OREIS	Oracle	DOE	Oak Ridge, TN	DOE site profiles
Basel	MS-Access	EEA	Austin, TX	Basel Convention transport data

means of a common ontology. The considerable structural correspondence among these legacy data sources is not by design, but because they address related problem domains. Consequently, they cannot be integrated using traditional methods, as the sources of a distributed database might be. Rather, independently-developed databases with dissimilar schemas and lexicons must be conceptually integrated in a dynamic manner. In these circumstances, value mapping and semantic translation, reasonably well understood in the context of schema integration, become dynamic problems that must be addressed in a flexible way. An agent-based system allows new functionality to be incorporated into an existing design, as well as allowing new or modified designs to be developed within the architecture of a functioning system.

The EDEN pilot demonstration concentrates on information relating to remediation of hazardous waste contamination. The ontology thus focuses principally on the relationships between contaminated sites, the wastes that cause the contamination, and technologies used to remediate specific kinds of contamination in specific media at each site. We are investigating the use of terms from the EEA's General European Multilingual Environmental Thesaurus[2] (GEMET) to create a standardized vocabulary and enable multi-lingual translation of the terms of queries and results.

For each database, a resource agent has been configured with mappings from the database schema to the common ontology. The system then provides access to all of these resources using the common ontology as its query framework. An especially beneficial effect of agent-based design is that a new resource can be brought into the system dynamically.

Several issues have become conspicuous in the course of the EDEN project. Because EDEN has many diverse types of users, it naturally requires a query interface

that can adapt to the needs of users of different backgrounds. Section 5.2 discusses how we allow declarative construction of useful parameterized queries over a high-level ontology.

Because the user interface has been abstracted from the normal level of database access, it becomes more difficult to understand the scope of a query. Neither database size nor even what databases may be accessed by a given query is readily apparent to the user. A seemingly innocent query can generate huge amounts of undesired data from several sources. Section 5.3 discusses reducing the risk of retrieving unnecessarily large results.

Value mapping among different representations of similar information poses significant challenges. Especially interesting are problems relating to differing granularities, such as in recording the presence of contaminants, which ranges from detailed scientific information on both sampling and analysis to mere inference of the presence of a contaminant at an actionable level.

Because value maps typically are useful either in multiple domains or are too complex or sophisticated to be addressed using traditional mechanisms, EDEN uses separate *value mapping agents* (see Section 7.2). In the EDEN pilot project, value mapping agents take advantage of the EPA's Environmental Data Registry (EDR), a reference implementation of the ISO/IEC 11179 meta-data registry standard.

The basic mode of interaction with EDEN is on-demand resolution of user queries. End-users query the system of agents with respect to the terms of the domain ontology. A user's query is solved by resolving query constraints with respect to all resource agents' advertisements and then querying only those resource agents that have advertised information that may satisfy the query.

MCC's Competitive Intelligence applications perform on-demand query resolution as well, but they also exhibit a more complex behavior mode based on continuing challenges, which are explained next.

3.2. Competitive intelligence

One of the services MCC provides is strategic, in-depth technical analysis of its member companies and their competitors, a practice known as *competitive intelligence (CI)*. Current practice in CI is characterized by manual information gathering – CI professionals tend to spend 80% of their time manually gathering data and only 20% of their time using tools to aid the gathering and analysis process. A particularly interesting and challenging application of InfoSleuth has been to use agents to reverse this 80/20 distribution to allow more time in high level analysis and hypothesis generation.

We have been using InfoSleuth agents to discover, acquire, integrate, and monitor CI information from open sources to provide on-demand historical snapshots of competitor statistics across multiple performance indicators; and to detect trends and changes in technology indicators for individual companies, for groups of related companies, or for general technology sectors.

Table 2. Types of competitive intelligence information sources

Source Category	Sample Sources
Macro-level statistics: sales, R&D expenditures, workforce profiles	SEC filings (www.sec.gov) Hoovers online (www.hoovers.com) company web sites
Current events	Press releases (www.prnewswire.com) CNN daily (www.cnn.com) company web sites
Growth technology	US, European, Japanese patent listings
Emerging technology	INSPEC publications database IEEE, ACM online bibliographies

InfoSleuth CI applications support continual *information discovery* using a set of resource agents and task planning agents that accepts domain ontologies as long-running "challenges" to discover related information. The resource agents encapsulate various levels of information analysis, from simple web crawling to more complex concept classification and extraction. The task planning agents organize data flow among these resource agents to accomplish a spectrum of information finding and analysis activities.

Each resource agent accepts subscriptions for long-running knowledge discovery tasks, catalogs the information it produces, and then advertises that information in terms of the domain ontology. As a result, a resource agent finds itself serving long-running information discovery challenges and then answering on-demand queries about news and trends with respect to the information it has produced for a given challenge. While the discussion in the rest of this paper focuses primarily on the problem of answering single queries, we discuss subscriptions and long-running tasks briefly in Section 9.

CI application activities are performed with respect to four types of information products, as listed in Table 2. Note that, for the most part, these resources are either collections of semi-structured text documents, or structured records containing various free-text attributes. As a result, technologies for extracting semantic concepts from text are of paramount importance in CI applications.

Each InfoSleuth CI application is characterized by a set of ontologies representing a technical definition of concepts and properties of a domain. The goal of each application is to *discover* information and information trends that match nodes in the domain ontology, *catalog* this information according to the ontology, and *inspect* the catalogned information and information trends.

Each domain ontology represents two things to the InfoSleuth agents. First, the particular domain description represented by an ontology is viewed by the agents as the terms of their knowledge discovery *challenge*. Second, the ontology gives a *representation* of the way the user would like to view and monitor a particular domain.

Two of the several CI applications are described here, technology tracking and intellectual property management.

Technology tracking: The goal of the CI Technology Tracking application is automatic creation and maintenance of a technology taxonomy, with documents classified as belonging to various nodes in the taxonomy. This application uses four types of InfoSleuth resource agents: those that interact with established information search services, those that discover and begin monitoring new information sources, those that extract context and content from source articles, and those that classify documents by extracted content.

In a typical interaction with this application, an analyst asks questions like "How many articles in May 1999 contained information about the price of active matrix displays? Show me the articles," or "How is the concept of 'active matrix displays' being discussed in the news? Show me the dominant terms and phrases associated with this concept."

Intellectual property management: The Intellectual Property Management application seeks to identify potential prior patent claims for a given patent or technical document. The user identifies a target document and highlights important "terms of art" in this target. InfoSleuth takes the target plus annotations as input and constructs a portfolio of potential prior claims. The sources of information include US and international patents, technical publications, general web searches, and focused company web site traversals. As with Technology Tracking, web site traversal and source monitoring are performed by one class of agents and detailed invalidation analysis by another.

4. Ontologies

In this and the following sections, we describe some of the functionalities and approaches we have taken within the InfoSleuth agent system for the various aspects of information gathering and analysis that we address. We begin with ontologies, which are foundational to our semantic approach to describing information.

4.1. What is an ontology?

Consider the query, "`Are computer hardware companies in the Austin area doing well?` " This query is problematic in that it is conceived using unrestricted natural language, which cannot be understood by InfoSleuth; furthermore it cannot be posed well in SQL, because there is no specification of either the source of the information or exactly what entities and attributes are being referred to by "computer hardware companies", "Austin area" and "doing well". Clearly, there is a need for some intermediate language that allows us to talk about things of interest on a high, resource independent level, would be straightforward for humans to use, and at the same time would be easier to deal with than unrestricted English. This language we call an *ontology*.

An ontology is a collection of concept types and their relationships, as used

in a domain of discourse by a specific group of agents (human, software, or both) towards solving a specific class of problems. When an ontology is used to integrate information, we can say that it constitutes a *semantic model* of the integrated information. It can then be used as a vocabulary for conversing about the integrated information and for describing it. There is no one ontology used in InfoSleuth, but rather a set of domain-specific ontologies, with one or more ontologies used for a given application. Because ontologies are domain-specific, it is easier to eliminate ambiguities. In InfoSleuth, each query is posed over some single, domain-specific ontology.

4.2. Representation of ontologies using OKBC

Ontologies can be formally represented in many different ways. Two well-known examples are the ER model with extensions for hierarchical knowledge, and the Horn-clause form of classical first-order predicate logic. One of the obstacles in integrating independently-developed ontologies is that of mapping different representational systems to each other. The database community developed ODBC to help address this issue. With similar motivation, the knowledge representation community has been working, under DARPA's coordination, on an analogous standard called Open Knowledge Base Connectivity, or OKBC.[3,4]

The OKBC protocol provides a set of operations for a generic interface to underlying knowledge representation systems (KRs). The interface layer allows an application some independence from the idiosyncrasies of specific KRS software and enables development of generic tools that operate on many knowledge representation systems. OKBC provides operations for manipulating knowledge expressed in an implicit representation formalism called the OKBC knowledge model. It supports an object-oriented representation of knowledge and provides a set of representational constructs commonly found in object-oriented knowledge representation systems. OKBC implementations exist for several programming languages, and provide access to knowledge bases both locally and over a network.

The ontologies used in the InfoSleuth system are formulated using OKBC, and this allows queries over ontological concepts to be formulated and used by the system independently of the knowledge base behind the ontology agent's OKBC server. Note that the adoption of a representational standard like OKBC does not solve the problem of *semantic* integration of the concepts in multiple ontologies– that is, the problem of how to determine whether two different objects in different ontologies refer to the same concept in the given domain of discourse. This is a much harder problem, and a planned area for future work in the InfoSleuth project.

4.3. Constructing and Modifying Ontologies

For InfoSleuth to be able to mediate and integrate information, it must be able to use, and provide tools for building and maintaining, one or more appropriate ontologies describing the domains of interest. The size and complexity of required

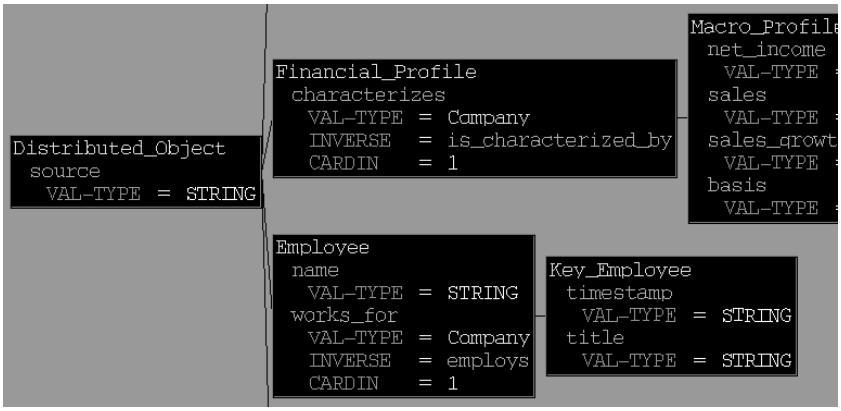

Fig. 3. Igor window showing a fragment of the Technology Tracking ontology.

ontologies varies greatly. For more complex domains, ontologies with hundreds and even thousands of concepts might be necessary.

We provide a graphical ontology editor, *Igor*, to help users build and modify InfoSleuth ontologies. Igor provides advanced features for constructing and visualizing large ontologies. Ontology fragments can be presented for inspection. A specialized query language has been developed which lets users formulate in a graphic and interactive way what ontology fragments they want to see. Some of the most sophisticated graphical browsing features rely on zooming (the use of internal cameras and lenses). Another device used in the graphical presentation of ontology fragments is the Poincare disk, invented originally for visualizing objects in hyperbolic geometry.

Igor also provides assistance to the user, and checks input changes to the ontology for internal consistency. For example, our editor prevents its users from specifying slot inverses that don't specify domain and range consistently. Similarly, taxonomical links which would cause circularity in the taxonomy graph are not allowed. When Igor detects that the user is about to create an inconsistency, it abandons the operation and displays an error message.

Figure 3 shows a screen with a view of a fragment of one of the ontologies used in the Competitive Intelligence applications. The box titled "Employee" defines a class for which two relations have been specified: "name" and "works_for". The latter relates this class to another class in the ontology (not shown), "Company". This relation, when viewed from the perspective of "Company" has a different name, "employs". The class "Key_Employee" is a subclass of the class "Employee".

4.4. Distributing Ontologies

Once an appropriate domain ontology is built, it can be deployed in the InfoSleuth system. *Ontology agents* wrap the OKBC servers that store the ontologies and provide them to agents that need them. They provide ontology specifications to

users for request formulation, to resource agents for mapping, and to other agents that need to understand and process requests and information in the application domain.

5. Querying InfoSleuth

5.1. Query language

The InfoSleuth query language comprises the main features of SQL2, plus some extensions from SQL3.[5] The main construct from SQL3 is the use of path expressions, which may appear in any of the *select*, *from* and *where* clauses. The syntax of path expressions is consistent with that used in both SQL3 and OQL.[6] Another feature of the language is that functions are allowed in the *select* and *where* clauses. The syntax of functions in InfoSleuth SQL is consistent with that used in relational database systems such as Oracle.

5.2. Query formulation

Queries in InfoSleuth are specified over ontologies. InfoSleuth provides a generic way of creating multiple viewing contexts over portions of a given ontology. This allows different users with different needs to access different kinds of data in different ways. The Template-based Query Markup Language (TQML) specifies a mapping between natural language query fragments and SQL over an ontology, and represents the parameters through entry fields or domain-valued menus and list boxes.[7] The TQML browser, which uses this mapping language, facilitates the specification of well-formed InfoSleuth queries. It also maintains a materialized view of the available data, derived from the results of queries that the user has specified as he looks through the data.

Query specifications are delivered to the browser by the User Agent. The user interface then populates choice lists from the user's locally materialized view of the ontology and uses the currently selected values to build the correct SQL query. Figure 4 shows the TQML query interface using the EDEN ontology and the "Scientist" viewing context.

Once the query is fully-specified, it is passed to the InfoSleuth agent system as an SQL query. InfoSleuth retrieves the requested information, and the results are returned in the form of Web pages. InfoSleuth annotates the result with the sites that responded with information. Figure 5 shows a result window.

5.3. Query restriction

In many applications that fuse heterogeneous information, database size and/or result size can become critical factors in the response time of a given query. This has a significant effect on the usability of such an information system. In particular, it affects the semantics of queries that can be considered acceptable to the system,

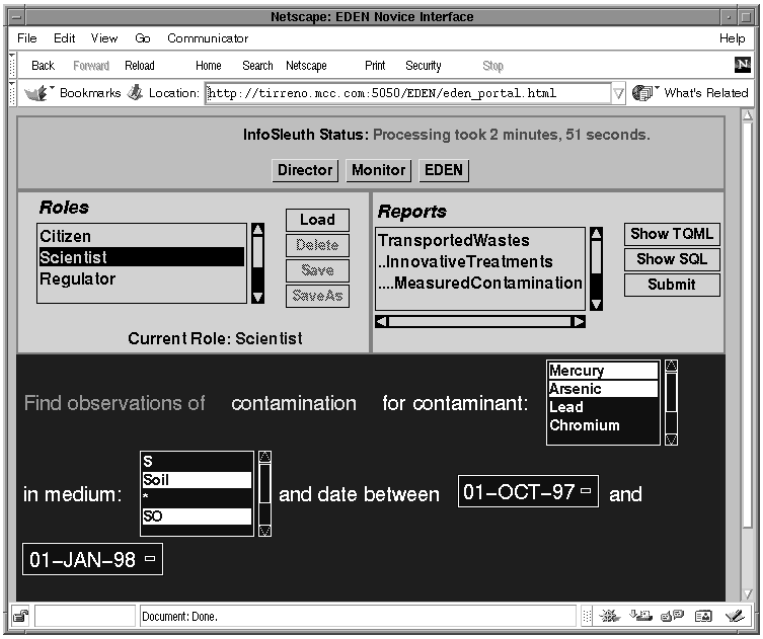

Fig. 4. Query window from the TQML browser.

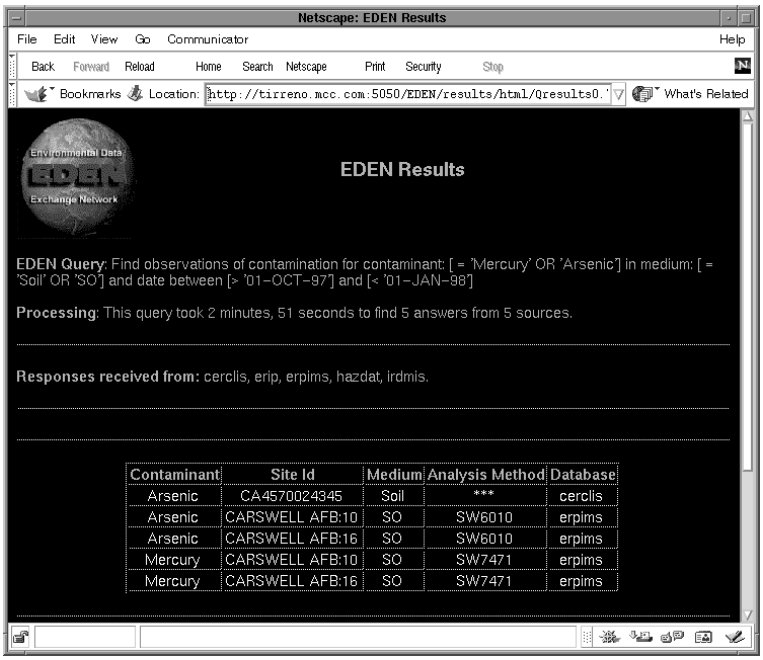

Fig. 5. Reply window from the TQML browser.

because inadequately constrained queries can result during the process of decomposing the query and physically mapping it onto information resources. Without addressing this issue in some way, no large-scale system can be viable.

We are developing declarative definitions of semantic constraints on classes in the ontology such that a user agent can discourage or forbid a user from posing a query on a particular class without specifying an adequate constraint in the `where` clause. Currently, this work is based on the query differencing operation described by Minock et.al.[8]

6. Brokering

Brokering is the dynamic location and recommendation of active agents relevant to a specific task. When a requesting agent has a specific task that needs doing, and it must locate some agent that can do that task, then it asks the broker to recommend specific agents to which it can forward its request. We describe brokering in more detail in a paper by Nodine et.al.[9]

6.1. Broker agents

The broker agent maintains a knowledge base of information that all the other agents have advertised about themselves. A broker agent implements the brokering service that uses this knowledge to match agents with requested services.

Multibrokering allows the process of matching service agents to requests to be distributed across multiple brokers, each representing a different set of agents. That is, brokers collaborate with each other in making recommendations to requesting agents for specific services other agents have advertised. When a broker receives a request for an agent with specific capabilities, it looks for matches in its own repository of agent information and may also query other brokers to find external agents with needed capabilities.

6.2. Advertising and querying

The brokering process follows an advertising/querying paradigm. When an agent comes online, it announces itself to some broker by *advertising* to it, using the terms and attributes described in the "infosleuth ontology" and constraints on the values of those attributes. The infosleuth ontology is a special ontology used by agents to specify advertisements and queries. Concepts represented within the infosleuth ontology include:

Content - What ontology subset is accessible by the agent?

Services - What can the agent do for you?

Syntax - What are the interfaces to those services?

Performance - How well can the agent do this at the moment?

Properties - Where is the agent? What protocols does it speak?

As is evidenced by the above list, the infosleuth ontology represents many different levels of characterizing an agent.

The broker stores all of the advertised information in its knowledge base. When an agent's set of available services changes, the agent may update its advertisement, and the broker will update the information in its knowledge base. When an agent goes offline, it unregisters itself to the broker, so that its advertisement can be removed from the knowledge base.

Agents requesting services formulate queries for servicing agents to the broker in terms of the infosleuth ontology. The broker then matches the request to servicing agents whose advertisements correspond to the constraints specified by the requesting agent. Finally, the broker returns a recommendation containing those servicing agents to the requesting agent.

6.3. The brokering process

The brokering process is the method by which an agent matches up a query with the agent advertisements to determine which agents can satisfy the specific request described in the query. As is evidenced by the different kinds of things represented in the infosleuth ontology, brokering occurs at many different levels of agent characteristics.

Syntactic brokering is the process of matching requests to agents on the basis of the syntax of the incoming messages. A classic example of syntactic brokering is found in CORBA (Common Object Request Broker Architecture)[10], which locates processes that can execute a method call that has a specific signature defined in its Interface Definition Language. In the agent-based community, KQML (Knowledge Query Manipulation Language)[11] also specifies syntactic-level brokering, as its advertise performative specifies one or more "performative forms" that the requesting agent can fill in when requesting a service from the advertising agent.

Semantic brokering is the process of matching requests to agents on the basis of the requested capabilities or services, and/or (in an information system) constraints on the information that an agent can provide.

Pragmatic brokering is the process of filtering a set of agents that provide the requested services and maintain the requested interfaces, in order to determine whether or not external considerations will affect the quality of service. Pragmatic considerations include such things as the performance of the machine the agent is running on and the security requirements of the agent with respect to the requester.

6.4. An example

For example, consider the following query specified in (extended) SQL over a company profile ontology.

```
SELECT name, products FROM technology-company
```

```
WHERE technology-company.site in Central Texas
```

At some level, the agent system will need to locate the SQL-speaking resources that contain the raw information needed to answer this query.

Consider that we have a set of resources that have advertised that they maintain company profiles, and another set of resources that have advertised that they have company site information, each advertising a select set of cities its information covers. The broker would first locate all resources that speak SQL in its repository (syntax). It would then filter through the resources to find those that understand the company profile ontology and have information on sites. It would filter through these to determine which ones, based on their advertised value constraints, could be in Central Texas (semantics). This is done using *constraint matching*. Note that the purpose of this last step is to eliminate obviously useless resources, and may possibly still select resource agents that do not have the intended information. From this remaining list, the broker agent may eliminate agents that do not have the correct access permissions (pragmatic).

A second query to the broker would cover company profiles for technology companies, and specifically within the profile, any resources that have the company names and products parts of the profile.

7. Information and Value Mapping

7.1. Resource agents and ontology mapping

Resource agents in InfoSleuth are responsible for bringing external information and external services into the agent network as "ontologically", or "semantically", annotated information resources. Every resource agent (RA) *wraps* some external information source or service. This wrapper:

- Identifies the select fragment of the overall domain ontology that the RA knows it can extract from its underlying information source.
- Advertises this ontological fragment that it supports, along with constraints on the instances it can provide.
- Advertises its querying capabilities (as a subset of InfoSleuth SQL), in addition to other capability information.
- Accepts queries against its advertised ontological fragment and produces information resources from its underlying source that map to facts in the ontology.
- Accepts subscription queries against its advertised ontological fragment and produces change notifications whenever the answer to the query changes in the underlying source.

```
Ontology:
  class Publicity(date, title, source, URL)
   subclass TechPub(tech-categories)
   subclass CompanyPub(company, topic)
Advertisement:
  class TechPub(date > '1998/12/01', source, URL,
       tech-categories in {'displays', 'imaging'})
```

Fig. 6. Resource agent advertisement example

To better understand the advertisements resource agents create, consider an ontology to consist of a set of classes and subclasses describing concepts in a domain. Furthermore, let slots (or properties) be attached to each class describing facts about that concept in the domain. Finally, assume slots are inherited along the class hierarchy. With this type of representation, a resource agent's advertisement would consist of three things: (1) the *most specific classes* in the ontology for which the agent has information; (2) the *subset of slots for each class* that the agent can provide information about; and (3) the *slot value constraints* the agent asserts to hold for the range of information it can provide about a particular slot. Consider the example in Figure 6. In this example we show a sample advertisement for concept `TechPub` such that the agent supports all slots on this concept except titles; and, furthermore, the agent asserts value range constraints on the date and tech-categories of the publicity information it can provide.

The InfoSleuth system provides a set of tools and Java templates for building resource agents with minimal or no programming. A growing set of standard RA types have been defined. Each of the standard RA types is attached to a new information source by merely specifying configuration and mapping files. Standard types include:

- JDBC RAs that encapsulate JDBC-accessible sources. JDBC is the Java DataBase Connectivity standard that provides uniform access to relational databases.

- Regex RAs that pull information from semi-structured text files based on grammars of regular expression matching.

- Text RAs that pull information from semi-structured text files based on focused syntactic and semantic language parsing.

- OODB RAs that encapsulate collection-oriented semantics and path traversals in object-oriented databases.

- Service RAs that encapsulate batch-oriented external services or tools and map ontological requests into scheduled invocations of the external service.

For all standard types, new RAs are built by specifying configuration files that define the behavior of the agent and its mappings between ontological concepts and local information source artifacts.

7.2. Value mapping

Ontologies specify canonical representations both of the concepts in the application domain and of value-domains for the actual domain elements. Data represented in other value-domains can be mapped into the ontology's canonical value-domain by both resource agents and user agents so that they may relate values expressed in the conceptual domains in the ontology to data as stored in real world databases and as perceived by users. The value mapping problem has several manifestations, including the following which are specifically relevant to the current InfoSleuth applications. In all of these cases, the actual process of mapping between value domains is independent of the shared ontology that the query is specified over.

- *Traditional:* Mapping takes place by imposing a view on that information and/or defining functions to translate the data from one value domain to another, as is currently done in relational databases, e.g. the work of Heimbinger and McLeod.[12] This type of mapping provides little support for semantics, but rather relies on the structure of the data.

- *Reasoning:* Sometimes the conversion of values requires sophisticated reasoning or computation. This might occur, for instance, if data is measured over intervals, but the duration and boundaries of the intervals differ between information sources. The reasoning approach is desirable especially when inferencing rules or computations are changing as new semantic knowledge is discovered.

- *Multi-ontology based:* When concepts in the common domain take their values from some set of external ontologies, mapping can be specified explicitly among the external ontologies as relationships between a term in one ontology and the related terms in each other ontology.

- *Changing equivalences:* The values of specific attributes may take one form from some (changing) equivalence class. This case differs from the previous one in that membership is not fixed (i.e., one value per value domain), but may be very flexible. This type of mapping often occurs with hand-entered data, where people may use different abbreviations or misspellings for the same item (e.g., "sulfuric acid", "sulph. acid", "sufluric acid"). Many of these equivalences may be derivable from multiple information sources.

Aside from the traditional approach, these methods for doing value mapping are fairly complex. InfoSleuth takes the approach of encapsulating the mapping of equivalent value domains in specialized value mapping agents. Users query and view

data in their preferred value domain, and their user agents access the value mapping agents to do the value mapping necessary to communicate with other agents in the canonical value domain. Resource agents whose databases do not maintain data in the canonical value domain access the value mapping agents to perform the converse mapping.

8. Query Processing

Query processing involves receiving a query, discovering which resources have the information requested by the query, forwarding fragments of the query to those resources, and fusing the results into a single result that answers the query. The focus of the query processing task is the (multi-resource) query agent. Given a query over some domain ontology, a query agent must decompose the query into

- a collection of subqueries, each addressed to a specific resource agent
- a collection of global queries that together serve to fuse the subquery results into an integrated answer to the query

The query agent must then coordinate the processing of the query, by sending out subqueries to their respective resource agents, collecting the results and executing the global queries. The issues in query processing are somewhat related to those encountered in multidatabase systems and mediator systems, [13,14,15,16,17,18,19] but there are some significant differences. InfoSleuth allows for resource agents to continually enter and leave the system, so the query agent does not have access to a global catalog of the data provided by each resource agent. Instead, it must collaborate with one or more brokers to identify the resource agents that can assist with the processing of a given query. Subqueries sent to resource agents are always expressed in terms of the domain ontology, issues of subquery translation and execution being encapsulated within the resource agent.

The interaction pattern for processing of multi-resource queries is shown in Figure 7 and described in detail in Perry et.al.'s agent interactions paper.[20]

8.1. Logical optimization

The query parser identifies the ontological classes that are used in the query, and a sequence of operations (in extended relational algebra) which can be used to generate the result of the query from the extents of the classes. Logical optimization is then applied, to yield a more efficient (but logically equivalent) sequence of operations. Typically this involves pushing select and project operations, so that they are applied first. The usual rationale for this is that these operations reduce the size of their operands, so applying them early serves to reduce the volume of data transmitted between agents. In InfoSleuth there is an additional benefit to be derived. By associating select and project operations directly with specific classes, the query agent is able to issue more specific requests to the broker, which can therefore more

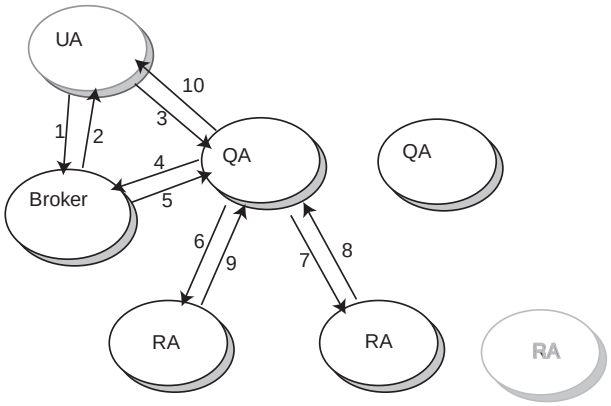

Fig. 7. Agent interaction pattern for query processing.

accurately identify the relevant resource agents. For example, if a query includes the condition "`company.city = 'Austin'`", and a resource agent has advertised that it knows only about companies in Houston, the broker will deduce that that resource agent is irrelevant to this particular query. Similarly, if the query involves "company.city" only, the broker would eliminate a resource agent that knows only company names and gross income.

8.2. Data localization

A query over a global ontology must be transformed into a query over the ontological fragments provided by specific resource agents. This requires the query agent to obtain, from a broker, details of those resource agents which have advertised knowledge of the relevant portions of the ontology. As described above, when a constraint and a subset of attributes can be associated with each class in the query, the broker is better able to eliminate irrelevant agents from consideration. A harder case is when a resource agent advertises some, but not all, of the attributes relevant to the query. In that case, the resource agent may still be used, but only if its data can be combined with those of another resource which provides the missing attributes. The query agent uses semantic analysis of resource agent advertisements to identify cases where two resource agents provide complementary information on the same entities. In such cases, the query agent introduces a join to combine the data from the two resources.

It is also necessary to take account of the fact that resource agents can advertise at different levels of the taxonomic hierarchy. Thus, it may happen that a query refers to a class "Company", while a resource agent has advertised knowledge of the class "TechnologyCompany". If "TechnologyCompany" is a subclass of "Company", the resource agent should still receive the query, which should be translated into the terms that the resource agent has advertised (i.e., "TechnologyCompany").

8.3. Query decomposition

Having determined which resource agents can provide a materialization of each of the ontological classes, the next step is to assign each operation to a resource agent. The aim here is to assign as much processing as possible to the resource agents which provide the data. However, we also need to take into account the capabilities of each resource agent. The query agent is itself responsible for executing those operations that the resource agents cannot implement, as well as those operations that combine data from two or more resources. The result of query decomposition is a collection of subqueries to be executed by resource agents, and a collection of global queries to be executed by the query agent.

8.4. Query execution

For a given decomposition, there are still several options available for processing a query. In particular, the execution can be optimized for all answers or for the first few answers.[21] When optimizing for all answers, the goal is for the query agent to as quickly as possible obtain all the necessary data for computing the result. Therefore, all subqueries are immediately sent out to the resource agents, and the query agent computes the result once all the resource agents have responded. This strategy works well when all subqueries return answers at about the same time. On the other hand, it is not very effective in an environment which includes a mix of fast and slow resources. In that instance, optimization for first few answers can yield substantial improvements in response time. The aim of that approach is to obtain sufficient data to compute at least some of the answers quickly, without having to wait for all the data from the slowest resources. The user will then receive the result as a stream.

Another technique which we have implemented for improving response times is to cache the results of queries to resource agents, so that subsequent queries can be answered from the cache without the need for costly access to slow resources. If a prior query Q_1 subsumes a subsequent query Q_2, the latter can be answered from the cache by means of a residual query Q_3 against the result of Q_1.[22]

9. Requests over Time

Subscriptions and periodic requests monitor a set of information, specified as a query, over time. In a periodic request, the user specifies a query and an interval, the agents compute the answer to the query at the beginning of each interval, and forward each result to the user. This implements a classic pull interface. In a subscription/notification, the user specifies a query, and the agents immediately generate a response to that query. When the answer to that query changes for some reason, the agents compute the specifics of the change and forward them to the user. This implements a combination pull/push interface, where the user tailors the query to his needs ("pull"), but the changes are sent back only as needed ("push").

9.1. Subscription execution

The subscription agent relies on the capabilities of the query agent in processing a subscription over time. It is an example of building one InfoSleuth functionality above a more basic one.

Figure 8 shows the agent interaction pattern for subscriptions. In the top left diagram, the user agent sends the subscription to the subscription agent. The subscription agent consults the query agent to obtain its query plan for the subscription content. Then, the subscription agent places subscriptions with the relevant resource agents based on the query plan. In the top right diagram, data in the resource agents has changed. The resource agents notify the subscription agent. The subscription agent then forwards the query to the query agent, which in turn queries the resource agents. In the bottom figure, the resource agents return their answers to the query agent, which computes the global query result and forwards the answers to the subscription agent. The subscription agent takes the difference between the new answer and the previous query sent to the query agent for that subscription. The subscription agent returns the computed differences to the user agent and caches the new result. Also in the lower diagram, asynchronously the subscription agent has requested a new query plan from the query agent and noticed that a new resource agent has come online, so it forwards a subscription request to the new resource agent.

9.2. Aggregation and filtering in subscriptions

InfoSleuth also supports the subscription retrieval of information at higher levels of abstraction and aggregation than are necessarily represented in the underlying data. Information is aggregated and abstracted using a variety of data mining and complex event detection agents which work in a cascading manner to process and digest the information into a level of abstraction appropriate to the user. Currently, our derived concept, deviation detection and sentinel agents, as well as our planned association rule mining agent, collaborate to accomplish these aggregation and abstraction tasks. These processes are described in more detail by Unruh et.al.[23]

10. Related Work

There are a number of other agent systems targeted towards information gathering tasks, which share similarities with InfoSleuth in the functionality of the agents as well as the system organization. Most contain agents which "wrap" resources and combine information using mediation techniques. These include SIMS[24], TSIMMIS[25] and InfoMaster[26,27]. These systems differ from InfoSleuth in that the actual translation of information from its local schema or representation to that of the ontology is done within the mediator itself. Because this translation and integration is done more centrally, these systems have yet to prove themselves when integrating even tens of agents. InfoSleuth's approach of having each resource

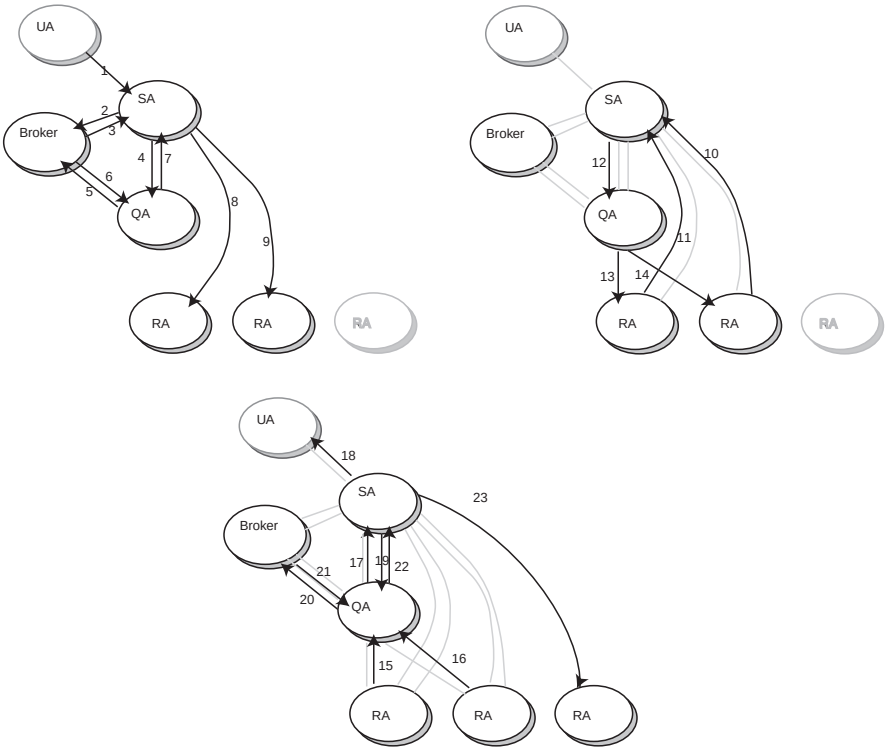

Fig. 8. Agent interaction pattern for subscriptions.

agent do its own translation to the ontological concepts has proven to scale to those levels.

The RETSINA agent framework[28,29,30] uses agents to address information retrieval in a similar manner to that of InfoSleuth. In RETSINA, "middle agents" provide the glue that links information sources to applications or users that need the information. RETSINA uses semantic brokering to locate required information; however, that brokering takes a different form than is used within our brokers. Furthermore, RETSINA has not been used for the same forms of complex query and subscription processing as InfoSleuth has been tested in. The process of using brokering or matchmaking to locate information for integration was also explored in the context of the SHADE project[31,32] and in OOA.[33]

Other work on using agents to gather and fuse information has been done by the government. These include DAIS[34] and the ARPI architecture.[35] These exercises so far have focused on specific tasks, and use a fixed set of agents to accomplish that task. It is not clear whether they can expand into the more flexible realm of dynamic agent communities.

11. Conclusion

In this paper, we have described InfoSleuth, an agent-based system for information gathering and analysis. The paper has emphasized InfoSleuth's ability to extract and advertise information about semantic concepts; to integrate information from heterogeneous sources; and to provide long-running information-gathering and analysis tasks. Software agents are used to dynamically couple these capabilities together to support monitoring and analysis of ontological concepts which change over time, at multiple levels of abstraction. Emergent from these capabilities are a set of goal-driven agent interaction patterns.

InfoSleuth is a deployed, advanced prototype system performing information gathering and analysis over open information sources. The success of our approach has been to realize that *real information gathering applications require a goal-driven interaction between information access, information integration, and information analysis technologies.* Whereas each of these technologies has received much prior attention, a flexible integration of these disciplines has not yet occurred. This has resulted in relatively few successful deployments of information gathering technologies in open information networks. A major component of our ongoing work is the active deployment of the system to distributed information gathering applications, such as EDEN and Competitive Intelligence. As these application deployments progress, we believe cooperating agent technologies will emerge as the proper technology to address these challenging, yet practical, application environments.

Acknowledgements

The InfoSleuth™ Project (http://www.mcc.com/projects/infosleuth) is an R&D project at MCC that is supported by various industrial and government sponsors.

In addition, we would like to acknwledge the hard work of the rest of the InfoSleuth team, including Richard Brice, Tony Cassandra, Damith Chandrasekara, John Dinsmore, Chung Hee Hwang, Qing Jia, Gale Martin, Mike Minock, Mosfeq Rashid, Marek Rusinkiewicz, Nancy Perry, and Bill Bohrer.

References

1. R. Bayardo et al. InfoSleuth: Agent-based semantic integration of information in open and dynamic environments. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 195–206. ACM Press, Jun 1997.
2. European Environmental Agency. General european multilingual environmental thesaurus, 1997.
3. V.K. Chaudhri, A. Farquhar, R. Fikes, and P.D. Karp. Open knowledge base connectivity 2.0. Technical Report KSL-98-06, Stanford University, 1998.
4. Richard Fikes and Adam Farquhar. Distributed repositories of highly expressive reusable ontologies. *IEEE Intelligent Systems and Their Applications*, 14(2):73–79, 1999.
5. J. Melton. A SQL3 snapshot. In *Proceedings of the International Conference on Data Engineering*, 1996.
6. S. Cluet. Designing OQL: allowing objects to be queried. *Information Systems*, 23(5), 1998.
7. M. Minock and J. Fowler. Template query mark-up language (TQML) 1.0. Technical Report INSL-122-98, MCC, 1998.
8. Michael Minock, Marek Rusinkiewicz, and Brad Perry. The identification of missing information resource agents by using the query difference operator. In *Proceedings of the International Conference on Cooperative Information Systems*, 1999.
9. M. Nodine, W. Bohrer, and A. Ngu. Semantic multibrokering over dynamic heterogeneous data sources in InfoSleuth. In *Proceedings of the International Conference on Data Engineering*, 1999.
10. Object Management Group and X/Open. *The Common Object Request Broker: Architecture and Specification, Revision 1.1.* John Wiley and Sons, 1992.
11. Y. Labrou. *Semantics for an Agent Communication Language*. PhD thesis, UMBC, Sep 1996.
12. Dennis Heimbinger and Dennis McLeod. A federated architecture for information management. *ACM Transactions on Office Automation Systems*, 3(3):253–278, Jul 1985.
13. D. Woelk, P. Cannata, M. Huhns, W. Shen, and C. Tomlinson. Using Carnot for enterprise information integration. In *Proceedings of the International Conference on Parallel and Distributed Information Systems*, 1993.
14. Y. Papakonstantinou, S. Abiteboul, and H. Garcia-Molina. Object fusion in mediator systems. In *Proceedings of the International Conference on Very Large Databases*, 1996.
15. S. Adali, K. Candan, Y. Papakonstantinou, and V. Subrahmanian. Query caching and optimization in distributed mediator systems. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1996.
16. A. Tomasic, L. Raschid, and P. Valduriez. Scaling access to heterogeneous data sources with DISCO. *IEEE Transactions on Knowledge and Data Engineering*, 10(5), 1998.
17. A. Levy, A. Rajaraman, and J. Ordville. Querying heterogeneous information sources

using source descriptions. In *Proceedings of the International Conference on Very Large Databases*, 1996.
18. M. Roth and P. Schwarz. Don't scrap it, wrap it! A wrapper architecture for legacy data sources. In *Proceedings of the International Conference on Very Large Databases*, 1997.
19. D. Florescu, L. Raschid, and P. Valduriez. A methodology for query reformulation in CIS using semantic knowledge. *International Journal of Cooperative Information Systems*, 5(4), 1996.
20. B. Perry, M. Taylor, and A. Unruh. Information aggregation and agent interaction patterns in InfoSleuth. In *Proceedings of the International Conference on Cooperative Information Systems*, 1999.
21. R. Bayardo and D. Miranker. Processing queries for first few answers. In *Proceedings of the International Conference on Information and Knowledge management*, 1996.
22. D. Miranker, M. Taylor, and A. Padmanaban. A tractable query cache by approximation. Technical Report MCC-INSL-127-98, MCC, 1998.
23. A. Unruh, G. Martin, and B. Perry. Getting only what you want: Data mining and event detection using InfoSleuth agents. Technical Report INSL-113-98, MCC, 1998.
24. Y. Arens, C.A. Knoblock, and W. Shen. Query reformulation for dynamic information integration. *Journal of Intelligent Information Systems*, 6(2):99–130, 1996.
25. Garcia Molina, Y. Papakonstantinou, D. Quass, A. Rajarman, Y. Sagiv, J. Ullman, and J. Widom. The TSIMMIS approach to mediation: Data models and languages. *Journal of Intelligent Information Systems*, 8(2):117–132, 1997.
26. D. Geddis, M. Genessereth, A. Keller, and N. Singh. InfoMaster: a virtual information system. In *Proceedings of the ACM CIKM Intelligent Information Agents Workshop*, 1995.
27. M.R. Genessereth, A. Keller, and O.M. Duschka. Infomaster: An Information Integration System. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 539–542, Tucson, Arizona, 1997.
28. K. Decker and K.P. Sycara. Intelligent adaptive information agents. *Journal of Intelligent Infromation Systems*, 9(3):239–260, 1997.
29. Katia Sycara, Matthias Klusch, Seth Widoff, and Jianguo Lu. Dynamic service matchmaking among agents in open information environments. *SIGMOD Record*, 1999.
30. Intelligent software agents.
31. McGuire, Kuokka, Weber, Tenenbaum, Gruber, and Olsen. SHADE: Technology for knowledge-based collaborative engineering. *Journal of Concurrent Engineering: Research and Applications*, 1(3), 1993.
32. D. Kuokka and L. Harada. Integrating information via matchmaking. *Journal of Intelligent Information Systems*, 6(2):261–279, 1996.
33. D. L. Martin, H. Oohama, D. Moran, and A. Cheyer. Information brokering in an agent architecture. In *Proceedings of the International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology*, April 1997.
34. DAIS Group. Dais. `<www.atl.external.lmco.com/projects/dais>`.
35. Donald McKay, Jon Pastor, Robin McEntire, and Tim Finin. An architecture for information agents. In A. Tate, editor, *Advanced Planning Technology*. AAAI Press, May 1996.

---

vice marks of their respective owners. All analyses are done without participation, authorization, and endorsement of the manufacturer.