Any Information, Anywhere, Anytime for the Warfighter

Mark B. Lazaroff and Philip A. Sage

Pacific-Sierra Research Corporation
1400 Key Boulevard, Suite 700
Arlington, VA 22209

ABSTRACT

The objective of the DARPA Battlefield Awareness Data Dissemination (BADD) program is to deliver battlefield awareness information to the warfighter – anywhere, anytime. BADD is an Advanced Concept Technology Demonstration (ACTD) to support proof of concept technology demonstrations and experiments with a goal of introducing new technology to support the operational needs and acceptance of the warfighter. BADD's information management technology provides a "smart" push of information to the users by providing information subscription services implemented via user-generated profiles. The system also provides services for warfighter pull or "reach-back" of information via ad hoc query support. The high bandwidth delivery of information via the Global Broadcast System (GBS) satellites enables users to receive battlefield awareness information virtually anywhere. Very similar goals have been established for data warehousing technology – that is, deliver the right information, to the right user, at the right time so that effective decisions can be made. In this paper, we examine the BADD Phase II architecture and underlying information management technology in the context of data warehousing technology and a data warehouse reference architecture. In particular, we focus on the BADD segment that PSR is building, the Interface to Information Sources (I2S).

Keywords: data warehouse, global broadcast system, information integration, intelligent software agents, reference architecture, information management technology, Zachman framework, InfoSleuth

2. CHARACTERISTICS OF A DATA WAREHOUSE

A "data warehouse" can be defined many ways, the definitions change as the technology base for data warehouses change. One common characteristic of a data warehouse is that it is a database application, with its own physical store, that ingests data from other databases and sources. These outside sources are usually operational databases and have often been designed as "stovepipe" systems that directly support user operations. Thus, the operational data that feeds a data warehouse is usually heterogeneous with respect to the data and the technology that implements it.

The operational data that supply the data warehouse may be collocated on the same machine or on the same local area network. Often the operational data may only be accessible via a wide area network. Therfore, a variety of data extraction and transport methods are used to extract data from the sources and move data to the data warehouse. When data are extracted data from multiple heterogeneous sources, they must be made uniform so that the data can be integrated inside the data warehouse. The transformation of the heterogeneous data into a uniform representation (i.e., a common schema or ontology) can take place at the source data repository or within the data warehouse.

Another characteristic of data warehouses is that the data contained within the data warehouse is subject-oriented. Subject-oriented means that the data supplied by the data warehouse are represented in a manner that supports the mission (subjects of interest) of the enterprise, is in a form that is familiar to the user, and conducive to its use. Depending on the enterprise to be supported by the data warehouse, data modes employing single or multiple ontologies are created for this purpose. The subject orientation affects the ontologies for a data warehouse in the level of detail in the data model. This approach is in contrast with process orientated approach in that the data warehouse data is designed to aggregate and summarize information at higher levels of granularity that are more conducive to the decision support requirements of the enterprise.

To support decision making requires that multiple data concepts and summarization of aggregate information be combined by integration of detailed data to create information. Information integration is the process of ingesting data and creating information tailored for use by the enterprise. The degree of difficulty of the integration is a function of the number and heterogeneity of the sources and the level of granularity or summarization required. Many heterogeneous sources can be integrated at low levels of granularity. A few heterogeneous sources can be integrated at high levels of granularity. Active research continues in integrating many heterogeneous sources at high levels of granularity. These challenges include:[1]

- data and metadata integration from multiple sources
- data cleanup and refinement
- data summarization and aggregation
- synchronization of sources with the data warehouse to ensure ongoing refreshment of the data warehouse as new data are created inside the sources
- performance issues related to sharing the same computer and database platforms as the data warehouse database and tools
- metadata management.

Another feature that distinguishes a data warehouse from an operational database system is that the temporal aspects of the data are contained within a data warehouse. Data warehouses are often designed to support historical analysis. The data within the data warehouse are time variant. Temporal variance also contributes significantly to the volume of data that the data warehouse must manage. The temporal aspects of the data warehouse are manifest in the system by:

- long time horizons (e.g., years) vs. short time horizons (days)
- data warehouse data is indexed by time
- frequency of database updates tend to be long.

The data warehouse is sometimes an operational data store. It delivers operational data to a wide range of users by making copies of the data from the operational database systems. In such a view, the data warehouse is required to effectively distribute operational data to a broad range of users. In this regard, the BADD system can expand the scope of a data warehouse by delivering data to a broad range of geographically dispersed users by way of the GBS satellite.

3. DATA WAREHOUSE REFERENCE ARCHITECTURE

3.1 The reference architecture

A number of key decisions must be made to design and build a data warehouse. These decisions are driven either by users (i.e., driven by the decision support requirements of the enterprise) or technology (e.g., demonstrating the potential of the technology to targeted users). Tradeoffs between the number of data sources to interface to the data warehouse and the level of information integration must be made. The data warehouse architecture must facilitate the engineering analysis by identifying the components that perform the following functions:[2]

- perform the extraction of data from operational systems
- clean and refine this data
- add time stamps and origin of the extended data
- structure and store the data to support a range of analysis tools and information analysis requirements.

Reference architectures are created to provide a conceptual framework of reference for various types of systems. The reference architecture serves as a set of rules and constraints for how a system will be designed and implemented. It provides a common method of communication or language for describing the components of the architecture and enables comparisons and engineering tradeoffs to be made. The reference architecture supports technology sourcing that can be made for the various components. This technology sourcing is the process of determining what components or parts of components will be purchased as commercial off-the-shelf products or must be developed.

At the highest level of abstraction, the data warehouse conforms to Wiederhold's Mediation Architecture.[3] The mediation architecture, shown in Figure 1 below, consists of three elements: (1) data, (2) mediators, and, (3) users. In this paradigm, the users are separated from the data, the data are heterogeneous in nature, and it requires mediation before the data are useable by the user. The data warehouse conforms to this abstract architecture because of the nature of the data, the mediators, and the users:

- The data in the data warehouse consist of a set of data extracted from multiple heterogeneous operational databases
- The mediators are required transform, integrate, aggregate, and summarize the data prior to access by users
- The users exploit the data and use it for analysis.

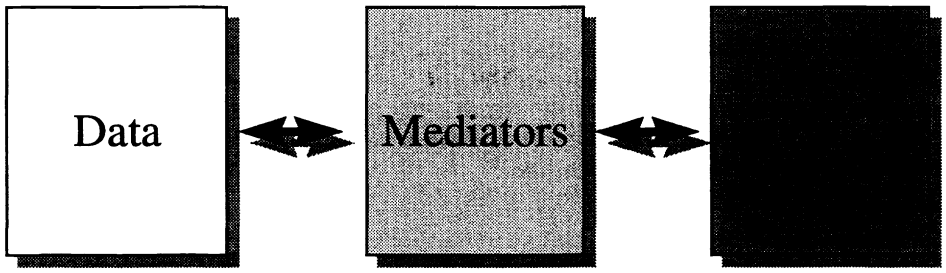

Figure 1 Wiederhold's Mediation Architecture

The data warehouse reference architecture and related descriptions, presented below, are based on the Zachman framework[4] and the Indica Data Warehouse Planner software system.[5] Additional details can be found in Gill and Rao.[6] Figure 2 below shows the top-level components of the reference architecture. The reference architecture decomposes the data warehouse into blocks and layers. Blocks represent specific data warehouse functions, and layers represent the environment needed to implement the blocks. The functional components are data source, data warehouse, data marts, and access and usage. The layers are data management, metadata management, transport, infrastructure, and tools, technologies, and roles.

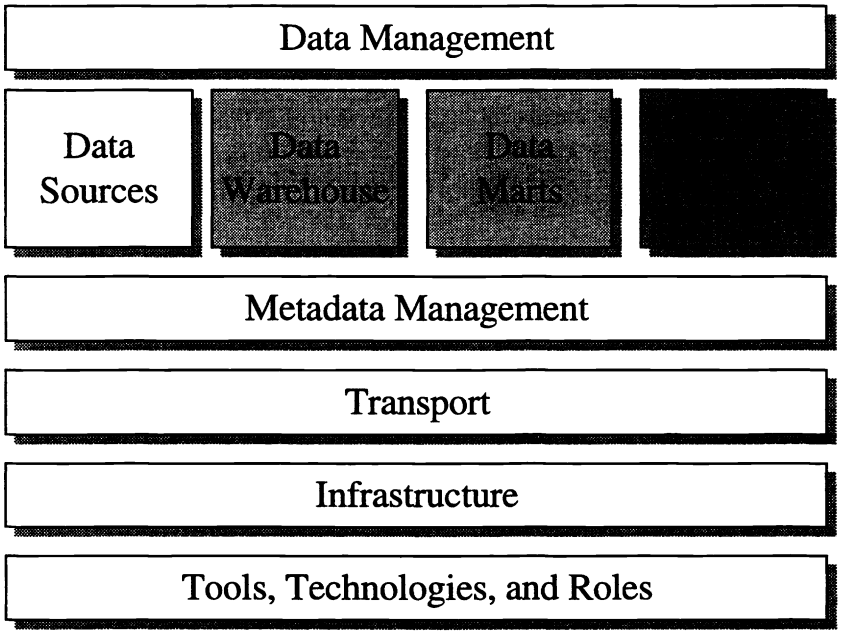

Figure 2 Top Level Components of Reference Architecture

3.2 Components

The components or blocks of the reference architecture correspond directly to Wiederhold's mediation architecture. The reference architecture data sources and access and usage components correspond to the data and user elements of the mediation architecture. The combined reference architecture data warehouse and data marts components combined correspond to the mediators of the mediation architecture.

The data sources component comprises all of the data source used to feed the data warehouse. This includes the structured database, unstructured data sources (e.g., free text), streaming data (e.g., video), etc. It also include the database catalogs or

metadata if such data exist. The data sources component may include transformation functions to map the source data to a common representation if this function is not centralized.

The data warehouse component includes all of the functions that are required to integrate data from the sources into information useable by the enterprise. These functions are categorized into three areas: 1) refinement, 2) reengineering, and 3) data warehousing. Refinement functions include standardizing, filtering and matching, cleaning up and scrubbing, time stamping and showing data source, verifying data quality, and metadata extracting and creating. Reengineering functions include integrating and partitioning, summarizing and aggregating, pre-calculating and deriving, translating and formatting, transforming and remapping, and creating metadata. Data warehouse functions include modeling, summarizing, aggregating, reconciling and validating, query building, creating glossary, and metadata browsing and supporting navigation. The design of the data model must be tightly coupled with the requirements of the enterprise.

The data marts component is a scaled-down version of the data warehouse. A data mart performs similar functions to the data warehouse except that it supports a smaller group of users.

The access and retrieval component provides functions for reconstructing data warehouse information into multidimensional views or for caching the information for data mining or other forms of analysis. Queries are initiated from this component and can take the form of profiles (e.g., standing queries or subscriptions) or ad hoc queries. Other functions of this component include metadata retrieval to support querying and browsing. Metadata are important because they provide a description of the contents of the data warehouse, the source of the data, temporal aspects of the data, and possibly any changes made in the integration of the data (e.g., conflicting data from multiple sources).

3.3 Layers

The horizontal layers correspond to the operation environment in which the various components, described above, operate. Both the data management and metadata management layers correspond to the processes of data extraction, loading, update and refresh that are needed to keep the data warehouse supplied with data. The other horizontal layers correspond to other common services that are available to the components. Usually the data source block and the transport and infrastructure layers exist prior to establishment of the data warehouse.

4. BADD IN THE CONTEXT OF THE REFERENCE ARCHITECTURE

4.1 BADD I2S Intelligent Agent Architecture

The BADD Phase II architecture conforms closely with Wiederhold's mediation architecture. The users are the Battlefield Awareness Applications (BAAs). The data information sources are extracted and streamed to the user via the GBS satellite. The mediator component corresponds to the Information Dissemination Server (IDS). The IDS provides advanced information management technology to provide the right information, to the warfighter, at the right time. The IDS consists of two segments, the Interface to Information Sources (I2S), and the Data Dissemination Server (DDS).

The I2S, depicted in Figure 3, is the BADD objective system's interface to information sources. These information sources are injected into the GBS broadcast stream. The information sources represent potentially widely distributed heterogeneous data sources that have local autonomy, no centralized indexing, and data characteristics that may vary widely as follows:

- data may be structured (e.g., relational databases),
- unstructured (e.g., free text), or
- streaming (e.g., Unattended video).

Characteristics such as latency, timeliness, and precision may also categorize the source data.

PSR is using Microelectronics and Computer Technology Corporation's (MCC) InfoSlueth[7] intelligent agent architecture to integrate these heterogeneous information resources. This approach is scaleable and is well suited for a dynamically changing environment such as the Internet, where sources may appear and disappear and registration of the sources may not be formally controlled. The system is able to dynamically determine information source status and to develop a pertinent query plan for satisfying the warfighters' information requirements. The system maintains update consistency among multiple information

sources though transactional workflows, carries out coordinated searches, and integrates retrieved data into understandable usable information.

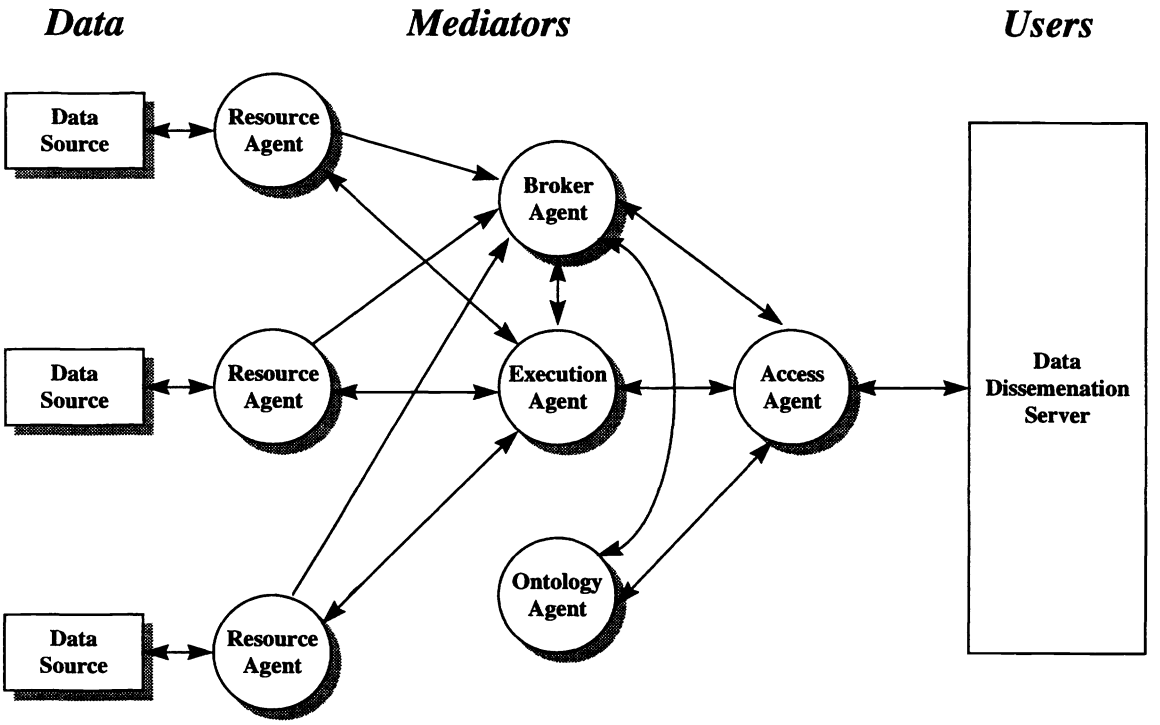

Figure 3 BADD I2S intelligent agent architecture

The access agent provides input into the I2S system and assists the user in formulating queries over a common domain model (i.e., common ontology or schema). At system start-up time the access agent advertises itself to the broker agent. The access agent obtains information from the ontology agent about the common ontological models known to the system, hence multiple ontologies are supported. These ontologies are passed to users and used to formulate queries or profiles. Queries formulated in terms of the selected ontology are sent to an execution agent that best meets user's needs with respect to current query context. When the execution agent obtains a query result, it engages in a conversation with the access agent in which results are incrementally returned. The user agent is persistent and autonomous, storing information (data and queries), and maintaining context between sessions. Explicit thread management in Java supports concurrent interaction with other agents. As system complexity grows, this capability can be used to maintain sets of applets as reusable modules in a warehouse separate from the access agent.

The ontology agent provides overall knowledge of ontologies and answers queries about ontologies. The ontology agent is a specialized resource agent that deals with the system's metadata. This agent receives queries from the execution agent for ontologies appropriate to the domain of a given query or profile. KQML is used for queries on the ontologies. Different ontology formats (e.g., relational, object-oriented, entity-relationship) can be described via ontology metamodels. Collaborating agents can request the list of ontologies currently supported or the entire contents of a named ontology.

The broker agent receives advertisements from resource agents about their services using a KQML performative "tell". Advertisements build up a set of metadata that describes the resources in terms of a common ontology or common schema.
The broker agent receives queries from access agents about the ontology (KQML ask-one, ask-all performatives) with an embedded KIF query specifying constraints to be satisfied. The broker agent returns the addresses of resource agents that satisfy the constraints. Replies to ask-one queries return a list of at most one tuple. Replies to ask-all queries return a possibly empty list of tuples, each of which contains the name and address of an appropriate resource agent.
The broker also receives "registrations" from access agents.

The execution agent coordinates execution of high level tasks or queries. It functions as a workflow manager. The execution agent performs global query decomposition into repository specific subqueries and performs postprocessing as subtasks. Global queries are executed in terms of the common ontology. Query plan and schedule generation is also a function of the execution agent. Query plan and schedule optimization is based on repository and profile/query specific constraints. An optimizer module generates an optimal query plan for sub-queries to be executed at various distributed sites. The execution agent also performs global joining of subquery responses. Plan execution is data driven and supports flexibility in reacting to unexpected events and incomplete information. For each query or profile (from access agents) a new plan is instantiated in the rule based system (CLIPS); multiple plans can be executed concurrently. Different execution plans can be made allowing synchronous or asynchronous operations and return of data to the access agent and the DDS.

The optimizer can support dynamic planning and scheduling of query plans. It has predictive modeling capabilities for how long a query response may take, thereby enabling scheduling under conditions of uncertainty. It can dynamically adapt scheduling based on query responses. This results in reprediction based on elapsed time and prior repository (resource) performance statistics. A logical extension of the idea is to incorporate a form of feedback mechanism. The idea is that as data are returned from queries, a to be determined component determines the value of the retrieved information to the model. Information value metrics are then passed back to the query planning components, stored as resource description data (metadata), and used to refine subsequent query plans and intelligent searches.

Resource agents are the interface to local data repositories. They run on the machine at the local repository. The resource agent provides services for information resource access, query translation, and communication. They provide a local view of the information contained within a repository in terms of the common ontology. The resource agent "advertises" the repository's scope in the context of the common ontology or schema (i.e., tables, attributes, domains, ranges) to a broker agent. The KQML agent communication language is used to send messages to resource agents. The resource agents respond to KQML and SQL queries. They provide monitoring capabilities over specified items in the local repositories using trigger and alert mechanisms

4.2 High level reference architecture view

Scope related decisions are represented by vertical segmentation of the reference architecture. A vertical slice through the system results in components that run on the same computer and share the same common operating environment. This is useful for planning decisions pertaining to, for example, whether the data sources and data warehouse should run on the same platform, whether there should be data marts, or whether the data warehouse and access and usage tools should be directly connected to the data sources. This segmentation of the reference architecture provides a physical view and can take into account requirements such as system availability, reliability, performance, and scalability.

Figure 4 depicts the reference architecture for the BADD Phase II System Integration Laboratory (SIL) demonstration. This configuration of the system shows that one data source, the Image Product Archive (IPA), will be located on a remote computer. It also shows that the second data source, the MIDB (an intelligence database), will be located on the same machine as the data warehouse and will share the same software infrastructure. In the BADD architecture, the data warehouse is the Information Dissemination Server (IDS). The IDS is "upstream" from the GBS broadcast, that is, it collects data from the data sources and pushes it to the GBS broadcast. The data mart component of the reference architecture is not part of the SIL demonstration system and is omitted. The access and usage component of the reference architecture is represented in BADD by compliant Battlefield Awareness Applications (BAAs). The BAAs are "downstream" of the GBS broadcast – that is, they are at the receive end of the system.

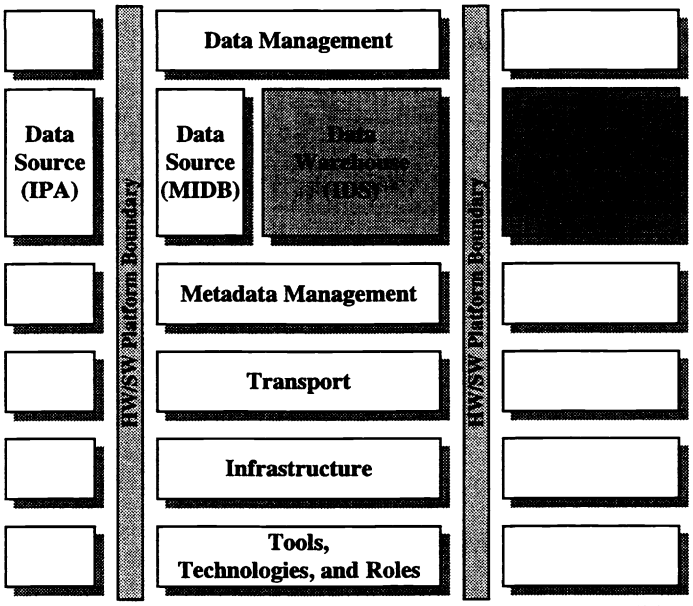

Figure 4 Segmentation of the reference architecture to locate HW/SW boundaries

4.3 Detailed view

A development view is also provided by the reference architecture. Here a detailed breakdown of the actual software module organization within the development environment is identified. Software is packaged into subsystems that are organized in a hierarchy of layers. Each layer has a well-defined interface. Figure 5 below, depicts a detailed decomposition of the data warehouse reference architecture. The decomposition is shown on the right side of the figure. The gray boxes indicate the I2S functions that map to the reference architecture elements.

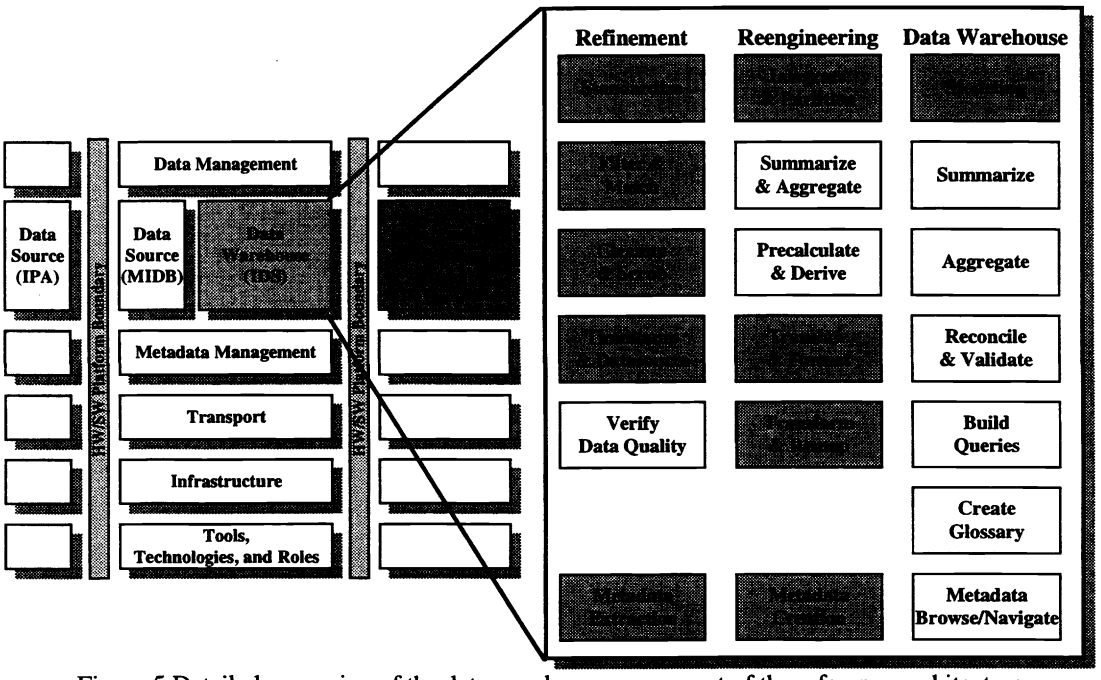

Figure 5 Detailed expansion of the data warehouse component of the reference architecture

5. SUMMARY AND CONCLUSIONS

The BADD I2S approach to multiple heterogeneous sources relaxes the constraint imposed by traditional data warehouse approaches by allowing the dynamic introduction and removal of information sources. The approach also supports the ability to dynamically model the different views or ontolgies of data by abstracting available information contained within the repositories and adding the user-view to the information. MCC's InfoSleuth agent architecture is used to integrate the heterogeneous information sources incorporating information about each repository. This information, known as metadata, is used to represent the following:

- the operating knowledge (e.g., trigger, process, and integrity constraints);
- control knowledge for system interaction, including data transfer rules and equivalence knowledge for data items; and
- decision knowledge such as decision rules and control procedures

for each of the sources.

The BADD I2S approach is generic and is being implemented to represent a modular system. Several research issues remain concerning the integration of information, how to represent distributed metadata, and transaction modeling remain.

ACKNOWLEDGEMENTS

This work was supported by the Defense Advanced Research Projects Agency, Information Systems Office. The opinions expressed in this paper are those of the authors and are not necessarily the opinions of the sponsors.

REFERENCES

1. H.S. Gill and P.C. Rao, *The Official Guide to Data Warehousing*, p. 4, Que Corporation, Indianapolis, IN, 1996.
2. Ibid. p.26.
3. G. Wiederhold, R. L. Blum, & M. Walker, *On Knowledge Base Management Systems: Integrating Artificial Intelligence and Database Technologies*, Springer Verlag, 1986.
4. J. Zachman, "A Framework for Information Systems Architecture." *IBM Systems Journal*, 1987.
5. Indica Data Warehouse Planner, Version 1.0, The Indica Group, 1996.
6. H.S. Gill and P.C. Rao, *The Official Guide to Data Warehousing*, 382 p., Que Corporation, Indianapolis, IN, 1996
7. InfoSleuth Release 2.5 Technical Overview, Microelectronics and Computer Technology Corporation, Austin, TX, Oct. 11, 1996.