Architectures and techniques for semantic data integration in distributed systems

CSC-30002: aDVANCED dATABASE SYSTEMS

Alex Farrell (15005594)

2017

Contents

[Table of Figures 1](#_Toc500865619)

[Introduction 2](#_Toc500865620)

[Section 1 – A Critical Review of Example Systems 3](#_Toc500865621)

[InfoSleuth – The Architecture and Its Limitations (550 Words) 3](#_Toc500865622)

[Key Objectives and Issues to Be Solved 3](#_Toc500865623)

[Data Resources Available 3](#_Toc500865624)

[Architecture and Concepts 3](#_Toc500865625)

[Use of Data Integration Methods, Semantics and Limitations 4](#_Toc500865626)

[SIMS – The Architecture and Its Limitations (550 words) 5](#_Toc500865627)

[Key Objectives and Issues to Be Solved 5](#_Toc500865628)

[Data Resources Available 5](#_Toc500865629)

[Architecture and Concepts 5](#_Toc500865630)

[Use of Data Integration Methods, Semantics and Limitations 5](#_Toc500865631)

[OBSERVER – The Architecture and Its Limitations 7](#_Toc500865632)

[Key Objectives and Issues to Be Solved 7](#_Toc500865633)

[Data Resources Available 7](#_Toc500865634)

[Architecture and Concepts 7](#_Toc500865635)

[Use of Data Integration Methods, Semantics and Limitations 7](#_Toc500865636)

[Section 2 – Generic Architecture of Heterogeneous Information Systems 9](#_Toc500865637)

[Section 3 – Modern Standards and the extraction of semi-structured data 11](#_Toc500865638)

[Semi-Structured Data – Uses and Drawbacks 11](#_Toc500865639)

[Modern Standards – Uses and Limitations 11](#_Toc500865640)

[XML 11](#_Toc500865641)

[JSON 11](#_Toc500865642)

[References 13](#_Toc500865643)

# Table of Figures

[Figure 1: InfoSleuth Architecture 3](file:///E:\3rd%20Year\1.%20Semester%201\Advanced%20Database%20Systems\1.%20Coursework\15005594.docx#_Toc500864132)

[Figure 2: SIMS Architecture 5](file:///E:\3rd%20Year\1.%20Semester%201\Advanced%20Database%20Systems\1.%20Coursework\15005594.docx#_Toc500864133)

[Figure 3: OBSERVER Architecture 7](file:///E:\3rd%20Year\1.%20Semester%201\Advanced%20Database%20Systems\1.%20Coursework\15005594.docx#_Toc500864134)

[Figure 4: Heterogeneous Architecture Representation 9](#_Toc500864135)

# Introduction

Modern database solutions are effective at handling large amounts of heterogeneous data on a single platform. However, it is more difficult to implement a solution that allows access to multiple databases or other information resources in a semantic, heterogeneous way that is understandable and easy to use. This essay focuses on the implementation of heterogeneous distributed systems.

Section 1 provided a critical analysis of some example systems based on research, that utilise both types of architecture, agent-based and mediator-based, discussing the problems each system looks to solve, as well as its architecture and limitations.

Section 2 discusses a generic architecture that can be used as a starting-point when implementing heterogeneous distributed systems, making use of concepts discussed in section 1 to show data processing methods from discovery to query execution.

Section 3 looks at modern standards such as XML and how they have been used to assist with integration and extraction of semi-structured information sources. This section looks at how they are used and what their drawbacks are.

# Section 1 – A Critical Review of Example Systems

## InfoSleuth – The Architecture and Its Limitations (550 Words)

### Key Objectives and Issues to Be Solved

The InfoSleuth project was developed between 1995 and 2000 by MCC. The building blocks for the project stem from the Carnot Project, also completed by MCC between 1990 and 1995. The goal of both InfoSleuth and Carnot is to address the problem of logically unifying physically-distributed, heterogeneous information (Woelk, et al., 1993) in a static environment. The InfoSleuth Project extends on this, making legacy database systems easily accessible on a dynamic, Web environment.

In terms of data integration, InfoSleuth focuses on locating, retrieving and networking information in a dynamic, ever-changing environment. Using new technologies (as of 1997), this information is merged so that it appears as a larger homogenous-like system (Bayardo Jr, et al., 1997).

### Data Resources Available

One issue highlighted with Carnot that InfoSleuth was developed to solve is that of the information sources accessible. Carnot was only developed to make use of information sources in a static environment. InfoSleuth successfully utilises information sources in a dynamic environment such as the World Wide Web (WWW). The WWW is an example of a publicly available networked resource, utilising the millions of live web servers located around the globe.

There is no defined data resource with InfoSleuth as due to the ever-changing nature of a dynamic heterogeneous environment. However, it makes use of both corporate and external networks such as those based on the National Information Infrastructure (NII) (Woelk & Tomlinson, 1994).

### Architecture and Concepts

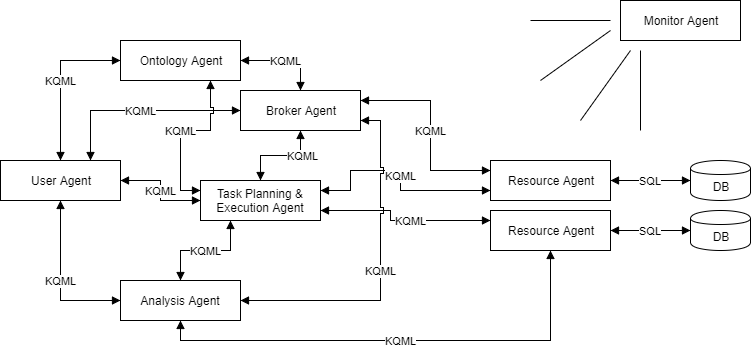
InfoSleuth is an agent-based heterogeneous system for discovering and retrieving information in a dynamic, open environment (Nodine, et al., 1999). These agents communicate with each other using Knowledge Query and Manipulation Language (KQML). KQML is a language used for exchanging information between knowledge-based systems and software agents.

Figure : InfoSleuth Architecture

InfoSleuth consists of several networked agents (See Figure 1) that allow user access through applets running on web browsers with Java capability, communicating with a personalized intelligent User Agent. (Bayardo Jr, et al., 1997). The User Agent uses knowledge of InfoSleuth’s ontologies (provided by the Ontology Agent), to help generate queries, displaying results from relevant information sources. The Broker Agent stores information on each agent, which is then used to respond to agent queries, routing them to information sources depending on the request. The Resource Agents and corresponding Data Analysis Agents provide mappings between the ontology, database schema and language of a resource, executing requests when necessary. The Task Execution Agent coordinates the gathering of relevant information sources necessary to respond to queries, using information from the Broker Agent, and reassembling the results. Finally, the Monitor Agent observes interactions between agents, providing an interface to display task execution.

### Use of Data Integration Methods, Semantics and Limitations

InfoSleuth operates under the concept of dynamic data integration. This allows for constant change and update of information sources with the integration of these sources being carried out using tools and special commands to create an integrated global view. Depending on the DDL and DML used in each system, the commands used may vary.

Dynamic integration also allows for scalability of information sources available to the system. Depending on the number of resources, there is an impact on decisions made with regards to the relevance of information to a query. In the case of InfoSleuth, these decisions can be made without direct access to the resource (Bayardo Jr, et al., 1997).

The ontologies used in InfoSleuth take the form of incomplete schemas. This means that although concepts can be arranged in a hierarchical class-based structure, no relations between these classes are permitted.

## SIMS – The Architecture and Its Limitations (550 words)

### Key Objectives and Issues to Be Solved

The SIMS project was developed in 1993 as a mediator-based system that provides access to multiple, integrated information sources. The goal was to provide a single, uniform way of describing information sources, making data more accessible.

SIMS deals with determining relevant information sources relating to knowledge extracted from user queries. If any knowledge-based class doesn’t have a matching information source, knowledge encoded inside the domain model (e.g. relationships), permit reformulation to identify suitable sources of information. SIMS also creates optimized plans using knowledge of information sources. Data-manipulation methods (e.g. subqueries) are created so that when they are executed, they help collate desired information (Arens, et al., 1993).

### Data Resources Available

Like InfoSleuth, SIMS uses dynamic information sources that are constantly changing and updating. Types of information sources used include databases, knowledge bases and other application programs (Arens, et al., 1996), with views of adding other resources such has HTML pages to this list.

As of 1996, the largest application domain acquired was a transport planning domain, containing information about movement of personnel and military equipment between locations by means of aircraft, ships, trucks, etc. (Arens, et al., 1996).

### Architecture and Concepts

A screenshot of a social media post

Description generated with very high confidenceSIMs architecture is made up of 2 main components (See Figure 2). The first component is the wrapper module which can translate a description of an information source in a given ontology, into an executable query that can be performed when requested. The wrapper also handles the communication between information sources.

Figure : SIMS Architecture

The other component is the mediator module. This unifies available information to appear as a larger homogeneous resource to the user, as well as providing the necessary language for access.

Together, these modules give SIMS the ability to intelligently retrieve and process data whilst removing the need for users and application programs to view locations, query languages and organisation of resources (Ambite, et al., 1995).

### Use of Data Integration Methods, Semantics and Limitations

Information sources are incorporated into SIMS by means of describing the data in the source that relates to the global domain model (global ontology) in the mediator. SIMS is classified as a single ontology approach to heterogeneous information systems (Wache, et al., 2001). The ontology is composed of classes and attributes extracted from an information sources such as the name and potential mappings to other information sources.

By using the global ontology, SIMS has the capability to evaluate the semantic heterogeneity of information sources, but only if the system has the same view on a domain. With regards to integration and deletion, the ontology needs to be modified to achieve this. Finally, a global ontology doesn’t allow for comparisons of multiple ontologies whereas a hybrid ontology makes this process simple using globally shared vocabulary that all information sources can be integrated to adhere to (Uzdanaviciute & Butleris, 2011).

## OBSERVER – The Architecture and Its Limitations

### Key Objectives and Issues to Be Solved

Observer is a wrapper, ontology-based system that proposes the management of multiple information sources through ontologies (Mena, et al., 2000). It looks at solving several problems to provide answers to queries. Resource discovery focuses on searching data repositories that are relevant to a submitted query. Also, different data repositories may have different ways of organising the data, as well as different formats and media types that may support different query languages. The way in which information sources are modelled that, although may contain the same information, may be modelled at different levels of abstraction. Other problems include the querying of information sources, problems with different vocabulary and situations where you will be dealing with information that isn’t precise (Mena, et al., 2000).

### Data Resources Available

OBSERVER deals with dynamic information sources. Unlike SIMS, it can handle more types of information such as HTML pages, databases and other files (Buccella, et al., 2010). The concept of data repository is used, which may be seen as a set of entity types and attributes. Each data repository has a specific organisation of its data and may or may not utilise a data manager (Mena, et al., 2000).

### A close up of text on a white background Description generated with high confidence Architecture and Concepts

Figure : OBSERVER Architecture

As already stated, OBSERVER uses wrappers and ontology servers to perform queries (See Figure 3). The wrapper module is used to understand the specific organisation repositories. This gives it knowledge on how to retrieve data from repositories that are hiding their data organisation.

OBSERVER is classified as a multi-ontology approach (Wache, et al., 2001). In this case, it constructs a model for dealing with multiple ontologies, whilst avoiding problems with integrating global ontologies.

Unique to OBSERVER is the Inter-Ontology Relationship Manager (IRM). This is a catalogue of system semantics specifically used to solve the problem with the heterogeneous vocabulary in different information sources is used to describe the same information (Buccella, et al., 2010).

### Use of Data Integration Methods, Semantics and Limitations

OBSERVER deals with dynamic data types. The implementation of the IRM enhances the scalability of the query processing strategy by avoiding the need for a global ontology. The reusability of information sources is also supported as OBSERVER defines local ontologies that might be defined for other purposes (Buccella, et al., 2010).

The absence of a predefined language means that different ontologies can be described using different vocabularies, depending on the needs of the user.

There is no available mechanism for the optimisation of submitted queries. Instead, the concept of ‘loss of information’ is used to describe the situation in which a user wants to expand a search query to include other ontologies. This method does benefit the optimisation of queries but not a great deal of time or processing effort is saved by doing so.

# Section 2 – Generic Architecture of Heterogeneous Information Systems

To produce a generic architecture that encapsulates the necessary components that make up a heterogeneous information system, it is important to understand the similarities and differences between different systems in terms of the methods used to process and integrate information, and the types of information each system deals with.

There are two categories that can be used to divide the types of information sources. The first is the State of the Information Source (SIS). This can be defined as either dynamic or static. When integrating dynamic information sources, the systems should implement mechanisms which identify which information sources are available at any given moment. The second category is the Type of the Information Systems (TIS). There are three main types of information supported: databases, files and HTML documents.

In terms of the architecture, there are two main types. The first is an agent-based approach such as InfoSleuth. This type of architecture uses agents that communicate with each other to exchange information between knowledge-bases and software agents (Buccella, et al., 2010). The other type of architecture is wrapper/ mediator-based. This architecture uses the idea of a global schema to provide access to information resources.

A close up of text on a black background

Description generated with very high confidence

Figure : Heterogeneous Architecture Representation

Figure 4 shows a representation of a heterogeneous architecture, incorporating concepts from both agent-based and mediator-based schemas. The architecture is predominantly based on OBSERVER, however there are concepts from both SIMS and InfoSleuth incorporated as well.

The idea of separating the user input from the query processor is because the language used will vary depending on the user. By implementing user input separately, you can allow for mechanisms to convert input into a specific language that can be used throughout the system.

Figure 4 shows the implementation of an IRM (Interontology Relationship Manager), used in the OBSERVER project. This module, viewed as a catalogue of semantics that is used to solve the problem of vocabulary, is a shared repository containing Interontology relationships (Mena, et al., 2000). Also taken from OBSERVER is the idea that the Information Sources, Ontologies and Mappings are fed into the Ontology Server which translates the user query into different queries to access the information resources. Each of these subqueries is expressed in the local query language used in the repository of execution (Mena, et al., 2000). The information retrieved is then returned to the user through the Query Processor.

From InfoSleuth the idea of an Execution Server is used. This gives the system the ability to coordinate and plan the execution of the query across the system (Nodine, et al., 1999). Also, the concept of a Data Analysis Server is used. This server creates mappings using meta-information from information sources and ontologies stored in the IRM. These mappings can then be fed into the Ontology Server, along with relevant Ontologies and information sources to produce results that, as described, are returned to the user through the query processor.

# Section 3 – Modern Standards and the extraction of semi-structured data

## Semi-Structured Data – Uses and Drawbacks

Semi-structured data is a form of structured data that does not follow rules of formal schemas associated with relational and object-oriented databases, which makes it self-describing (schema-less) (Buneman, 1997). With reference to dynamic data sources used in most heterogeneous systems, it is possible that the structure may change unpredictably.

Semi-structured data allows programmers to side-track the idea of object-relational impedance mismatch (conceptual and technical difficulties often encountered when a relational database management system (RDBMS) is being served by an application program). Semi-structured schemas also allow for a flexible format for data exchange between various heterogeneous databases.

## Modern Standards – Uses and Limitations

Modern standards have led to different and more improved integration and extraction methods for semi-structured data, which also helps with the interoperability in heterogeneous systems.

### XML

XML was developed in 1996 by W3C as a new standard for exchanging data over the web (Bray, et al., 2006). XML is a text-based mark-up language, like HTML, and is also a subset of SGML (Standard Generalised Mark-up Language). XML is a standard for data communication within the software industry and it is a primary way of data interchange amongst businesses. Ultimately, XML is likely to become a language used when creating most documents stored both on and off the internet.

XMLs specification includes a set of rules for designing semantic tags. Due to this, XML can be regarded as a meta-mark-up language that can be used to define other languages. Authors of XML documents can create custom elements which they can then use to represent their data sets. It is also possible to query XML documents (Sahuguet & Azavant, 1999) using logical or mathematical expressions, as is the case in a conventional database system. The idea of being able to query an XML document, helps with the extraction of information from semi-structured schema’s as, like meta-data in a HTML document, you can provide information about the document such as author, keywords, etc.

The extraction of data, transformation and data integration are issues that are well documented in database schemas that rely on a query language. However, common query languages such as SQL (Structured Query Language) and OQL (Object Query Language) cannot be used directly to query XML data, due to the flexibility and irregularity of XML-based documents.

### JSON

JSON or JavaScript Object Notation, is a lightweight, text-based, language independent data interchange format, derived from the ECMAScript Programming Language Standard (Crockford, 2006). JSON uses text to transmit data objects that consist of pairs of attributes. This allows it to transmit data between a server and a web application, like XML. Databases such as MongoDB and CouchBase, store data natively in a JSON format, allowing the full integration of the advantages associated with semi-structured data.

JSON is supported inside JavaScript, providing significant performance gains compared to XML which requires extra libraries to retrieve data from domains. It is estimated that JSON will parse more than 100 times faster than XML when used with modern browsers (Nurseitov, et al., 2009).

However, JSON does not support namespace and have a lack of input validation and extensibility. There is also no support for formal grammar definition, meaning that it is hard to communicate an interface to a user. Whereas XML has support of a wide array of development tools, JSON is very limited when it comes to this (MyArch, 2007).

# References

Ambite, J.-L.et al., 1995. *The SIMS Manual Version 1.0,* s.l.: USC/Information Sciences Institute.

Arens, Y., Chee, C., Hsu, C.-N. & Knoblock, C., 1993. Retrieving and Integrating Data From Multiple Information Sources. *International Journal on Intelligent and Cooperative Information Systems.*

Arens, Y., Hsu, C.-N. & Knoblock, C., 1996. *Query Processing in the SIMS Information Mediator.* Marina del Rey, Information Sciences Institute and Department of Computer Science Universiy of South California.

Bayardo Jr, R. J. et al., 1997. InfoSleuth: agent-based semantic integration of information in open and dynamic environments. *ACM SIGMOD Record,* 26(2), pp. 195 - 206.

Bray, T. et al., 2006. *Extensible Markup Language (XML),* s.l.: W3C.

Buccella, A., Cechich, A. & Brisaboa, N., 2010. *Ontology-Based Data Integration Methods: A Framework for Comparison,* s.l.: s.n.

Buneman, P., 1997. *Semistructured Data,* Philadelphia: University of Pennaylvania.

Crockford, D., 2006. *The application/json Media Type for JavaScript Object Notation (JSON).* [Online]   
Available at: https://tools.ietf.org/html/rfc4627  
[Accessed December 2017].

Mena, E., Illarramendi, A., Kashyap, V. & Sheth, A., 2000. *OBSEREVER: An Approach for Query Processing in Global Information based on Interoperation across Pre-existing Ontologies,* Boston: Kluwer Academic Publishers.

MyArch, 2007. *JSON Pros and Cons.* [Online]   
Available at: https://myarch.com/json-pros-and-cons/  
[Accessed December 2017].

Nodine, M., Bohrer, W. & Ngu, A. H. H., 1999. *Semanitc Brokering over Dynamic Heterogeneous Data Sources in InfoSleuth.* s.l., s.n.

Nurseitov, N., Paulson, M., Reynolds, R. & Izurieta, C., 2009. *Comparison of JSON and XML Data Interchange Formats: A Case Study,* Bozeman: Montana State University.

Sahuguet, A. & Azavant, F., 1999. *Looking at the Web through XML glasses,* s.l.: s.n.

Uzdanaviciute, V. & Butleris, R., 2011. *Ontology-based Foundations for Data Integration.* s.l., s.n.

Wache, H. et al., 2001. *Ontology-Based Integration of Information - A Survey of Existing Approaches.* Brehmen, University of Brehmen.

Woelk, D. et al., 1993. *Using Carnot for Enterprise Information Integration.* s.l., s.n., pp. 133-136.

Woelk, D. & Tomlinson, C., 1994. *The infosleuth project: intelligent search management via semantic agents.* s.l., Second International World Wide Web Conference.