Weather Data Warehouse: An Agent-Based Data Warehousing System

Gunjan Kalra
Donald Steiner
Quantum Leap Innovations
{gk, dds}@quantumleap.us

Abstract[1]

The quality and content of data regarding certain types of information can vary over time. Such data may be weather data, news reports, demographic data, and health data. To effectively utilize such data, it has to be gathered, processed, stored, and provided to information consumers in different formats. In case of different sources, the data must also be processed taking different formats and update frequencies into account. An agent-based system supports deploying and provisioning data processing services where they are best suited – at the information source, data warehouse, broker, or information consumer. We describe an agent-based approach to data warehousing enabling dynamic and seamless integration of new information sources and data formats. The system provides data to the ultimate information consumers on a subscription (push) or request (pull) basis. We have applied this architecture to provide a dynamic and flexible weather data warehouse that provides a wide variety of weather data from different sources to different weather-based applications.

1. Introduction

Retrieving data usually involves constraints with respect to knowledge about structure and semantic content of the data, location of the data source on the network, etc. The task of accessing data requires querying the source for the data, validating it, and translating it to the required format along with including any missing values. As the number of data sources increase, there is also an increase in the complexity of storing corresponding meta-data about available data and the data sources.

Certain types of data, such as news reports, demographic data, and weather data, may continually change over time. Furthermore, this data may have different sources and formats that also change from time to time. Information coming from different sources tends to vary in terms of not only data quality and content but also data format and update frequencies. For example, news articles differ in content and quality over a period of time, across different newspapers and websites. Moreover, several news providers may write articles on different aspects of the same incident. News gets validated typically anywhere from a few hours to days.

Another example of such time-variant data is weather data. The U.S. National Weather Service (NWS) [26] provides non-validated forecast data (Figure 1) whereas the U.S. National Climatic Data Center (NCDC) [24] provides validated historical data (Figure 3). Other than focusing on different aspects of data, the various sources may supply inconsistent values for the same data, while claiming higher quality in terms of completeness accuracy, etc. Weather Services such as Weather.com [34], AccuWeather [4], and the National Weather Service [26] provide potentially contradictory weather forecasts for the same locale. Figures 2 and 3 show data existing in different formats at the same source.

A routine day for a weather expert revolves around collecting current weather data from a standard weather forecasting source. If needed, the expert retrieves weather data for past days from another source. Since these services provide their data either by a URL-based query interface or by semi-structured text/HTML files, the expert has to retrieve data by looking through web pages or querying across all the required locations. At times, locations are designated by zip codes, which requires 99 different queries for the state of Delaware (one of the smaller states in the U.S.). Following the collection of data in source-specific format, the expert may be required to convert this data into a standard format after discarding source formatting and extracting the actual data. Also, efficient storage in the form of caching data is required when the sources have transient or volatile data or to avoid duplicating efforts for future access by applications that require weather data.

Furthermore, as forecast weather data is validated (typically within 2 or 3 days), the expert must update the repository accordingly. Also, the expert is responsible for

[1] This work was funded in part by the United States Office of Naval Research under Contract N00014-02-C-0320.

providing weather data to the relevant applications in the required formats. If the applications also require advanced meteorological data such as radiosonde (upper air) data, the expert must integrate weather data from the different sources per location and date.

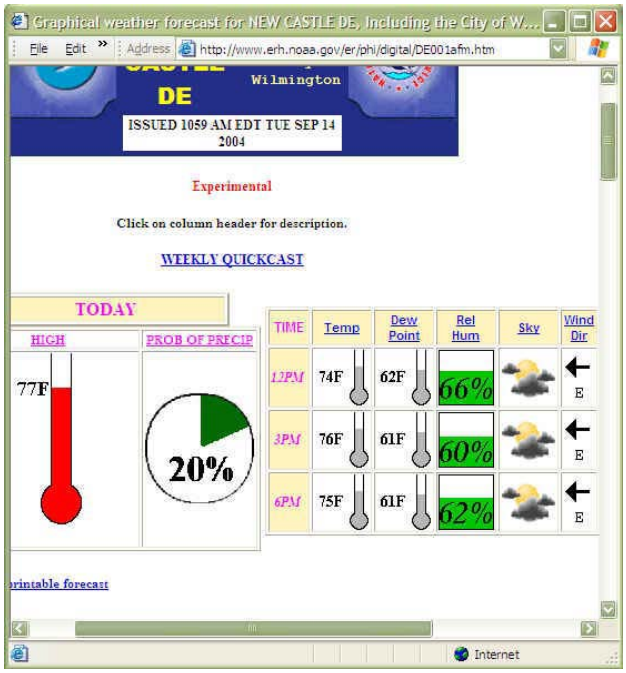

Figure 1. National Weather Service html-based forecast data

Data that is static and consistent across sources and over time can be easily retrieved directly from the data sources. But in the above situation, where tools and applications (consumers) require time-variant data, new system components must be incorporated to perform a series of processing steps on the data to ultimately make it very useful. Consumers must maintain their own enormous databases in cases where the providers only store the data temporarily (non-persistent, or transient storage). The ideal solution is a logical value chain with different components focused on providing the services required for handling time-variant information. In this paper we describe such a system.

Section 2 provides an overview of related work in the area of information integration. Section 3 highlights the typical data-processing services. Section 4 gives a brief overview of software agents. Sections 5 and 6 describe our system architecture and its application to the weather domain, respectively. Section 7 describes a sample enhancement of the weather application. Sections 8 and 9 outline directions for future work and the conclusion, respectively.

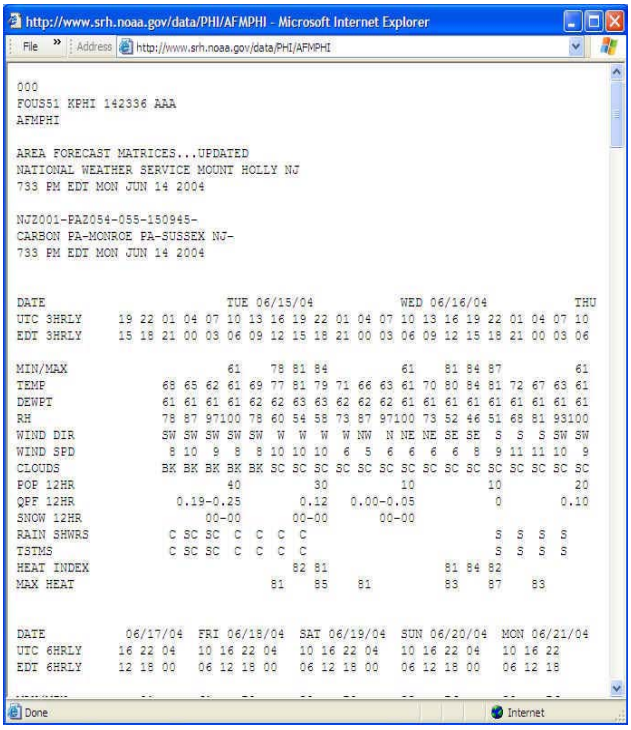

Figure 2. National Weather Service text-based forecast data

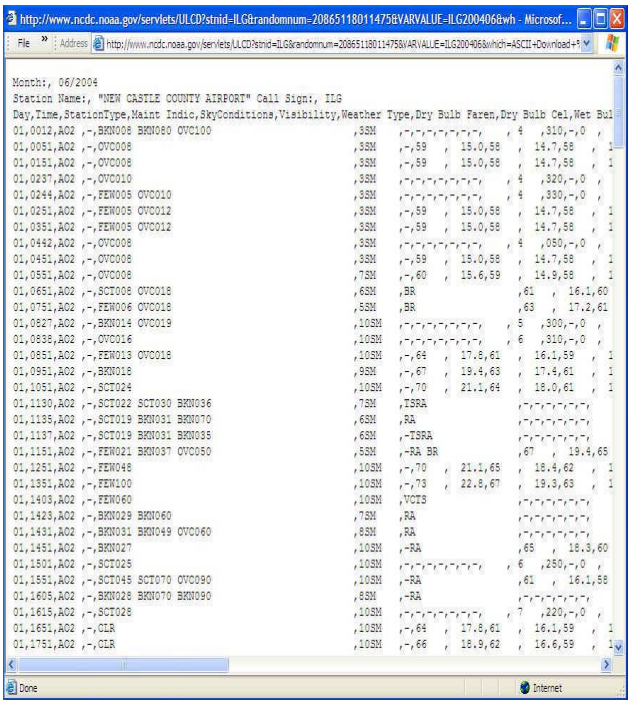

Figure 3. National Climatic Data Center text-based validated historical data

2. Related Work

Initial research in the database community that addressed the issues of information integration was directed towards federated databases [3]. More recently and more widely adopted is the 'data warehouse' approach to data acquisition, storage, and distribution that provides a central repository of historical, non-operational data to support management decision making by allowing data analysis over time. Although data warehouses store data essentially from a business analysis perspective, they face similar challenges of data consolidation. Widom [15] outlines the various research issues related to the design and implementation of data warehouses. The facts that data warehouses have a predetermined set of data sources, from which they gather data, and that they inherently have complete control over the data filling up the repository, makes the task of translating and integrating data very straightforward. However, adding a new source not only requires bringing down the system temporarily for configuration updates, but also necessitates incorporating consolidation and verification rules for the new source. There have been efforts towards definition and maintenance of data warehouses using materialized views (WHIPS) [32] and their combination with virtual views (H2O) [12]. Researchers have also focused on applying the mediator architecture to information integration systems [11]. The mediator-based approach to heterogeneous information integration and querying is also known as 'on-demand' approach since a source is integrated only when there is a query for the information it provides.

Two assumptions made by the above traditional approaches to integrating heterogeneous data sources are that all the information sources to be integrated are known a priori and that there is a uniform notion of semantics across the sources.

For addressing the assumption of uniform semantics, many wrapper-mediator style systems adopt either a predefined global schema [20] [33] or construct one based upon the source schemas [30], in order to describe the information and their sources. Wrappers interpret the local information about their information sources in terms of the global schema and a mediator handles the user queries, sends them to the appropriate wrapper, and consolidates the individual results. Several methodologies applied in such architectures include a Common Thesaurus in the MOMIS [30] project, description logic in the Information Manifold [1], annotation of object models and view construction in TSIMMIS [31], knowledge representation techniques in the SIMS [1] and the InfoMaster [20] efforts, AI planning techniques in the OCCAM [5] project, and highly structured relational models in the WHIRL [33] effort. Particularly for web information sources, learning by induction has been applied in order to create wrappers, including the work by Kushmerick [22] and Muslea (Stalker) [14]. These wrapper induction systems apply machine learning algorithms to learn web resource extractors by using manually labeled training data sets. In our system, we focus more on the overall optimized efficiency of the system than on that of the individual agent.

All the above efforts do not take into account the fact that new and external providers of certain types of information make significant contributions to the data repository and, hence, need to be dynamically integrated into the system. Thus, focusing on the first assumption of fixed information sources, we draw upon the concept of intelligent software agents [17] [27] [21]. (We provide an overview of software agents in Section 4.) Researchers have implemented several architectures for information integration system by using the intelligent agent paradigm. Decker et al [16] discuss 'middle agents' that provide the service of mediating between information producers and consumers. Consumers of information specify their preferences for data such as data format and data update frequency. Producers of information may advertise advantages of the data they provide, such as their data being most recent. Decker et al mention 9 categories of middle agents based upon the knowledge of consumers' preferences and providers' capabilities. In the 'broker middle agent' architecture, the providers and the consumers do not know the preferences and the capabilities, respectively, but the broker is aware of both.

In the above architecture, the broker agent focuses on solving the connection problem between consumers and providers of data. In case the consumer's preferences are not met completely, the broker agent is required to solve the same connection problem for each of the remaining services needed by the consumer. For example, if the only available information producer does not provide the data in the desired format, the consumer may end up looking for a provider of an appropriate translation service. Of course, to be able to find the translation service, the consumer has to have knowledge of the format of the data being provided by the producer. Furthermore, in case producers only have partial information, the consumer has to either gain knowledge of information-specific rules for merging the components or look for an agent that can provide this information fusion service. An example of a 'broker middle agent' architecture is Napster [23] that has a distributed, decentralized architecture of information providers and consumers along with a centralized directory. Napster's central index server maintains a list of connected clients and indices of their shared information. Also, every information provider is responsible for maintaining a repository of files that they provide as well as updating the central directory with the indices (filenames) of these

files. Consumers receive information from different providers and perform information consolidation as per the information domain. As the demand for information quality increases, the integration logic layer at each of the consumer nodes gets more complex. Moreover, information such as music files may not be available in the required formats. Unless components are available to translate the files into suitable formats, they are of no use.

Example agent-based information integration efforts are RETSINA [18], MIKS [29], KRAFT [2] and InfoSleuth [28]. InfoSleuth has a similar architecture as ours in terms of the type and roles of the agents. They do not explicitly describe incorporation of web sources that change frequently in terms of information layout and content. Also, they do not provide for caching frequently queried information – this may result in unnecessary communication and overloading of their resource agents. In our system, the data warehouse reduces most response times to a single lookup in the repository. InfoSleuth focuses more on one-time querying by users for data. By offering complete data warehousing functionality, our system provides complete and consistent data to the users beyond that which is solely available at the source. InfoSleuth describes a few fixed, yet complicated, interaction protocols for communication of data across the various agents. Our system is able to dynamically adopt more efficient protocols, according to where the data is actually stored. Also, InfoSleuth provides only a web interface for human users, whereas our system provides an agent wrapper API that can be easily used by other software applications.

3. Data-Processing Services

Information sources provide data in varying formats such as HTML pages, emails, plain text, etc. Before the concerned applications can efficiently utilize this data, they must apply a series of processing services to the data. A significant set of such services are mentioned below.

Retrieval: This service connects to the data source and captures the data as it exists at the source with source-specific formatting. For example, this service would post a query at the weather.com website and return the resultant web page (HTML).

Parsing: This service extracts the required data from the source-specific data as returned by the retrieval service. For example, this service would extract the value of the current temperature from the weather.com web page, based on the attributes embedded in the HTML description.

Standardizing: Data exists in different formats with different sources. Parsed data must be converted to a standard storage-efficient format for future data requirements and querying. For example, this service would parse the weather data fields' values into java objects that are easily updated and maintained.

Validation (Updating and Completion): Data validation involves applying domain specific rules to ensure data correctness. As per LabCompliance's Glossary [19], data validation is "A process used to determine if data are inaccurate, incomplete, or unreasonable. The process may include format checks, completeness, checks, check key tests, reasonableness checks, and limit checks." Some examples of this service are checking if -1000 degree Fahrenheit is a valid value for temperature, updating today's weather (available from the forecast service) with historical data when it is available and replacing the missing value in the temperature field by the standard value (-273 degrees Fahrenheit).

Consolidation: This service integrates data from different sources into the same repository. As per British Columbia Government's Information Resource Management's glossary [13] data integration is the process of 'blending data items from various distinct sources to create a larger and more comprehensive body of knowledge.' For example, historical and forecast weather data as retrieved from their sources is stored in one repository.

Storage: This service allows long term access to data in case of individual data sources having transient data stores. This service also allows storing different components of the data at one location.

Formatting and Translation: This service includes converting data into application-specific formats, e.g. GIS applications may require data in the form of GIS layers whereas traffic monitoring tools may require the same data in the text form.

These services can be combined in numerous ways resulting in different architectures that can be utilized as per the system's requirements.

4. Software Agents

An agent is an autonomous software program that performs a set of tasks (in our case, provides services) and can find and interact with other agents in its environment. Agents exhibit goal-directed behavior and respond to changes in their environment. Agents can perform tasks individually or work cooperatively in teams. Agents may also migrate between machines by virtue of being a complete unit of execution that can make its own decisions. Agents communicate with each other and with other machines via messages. Agents can be enhanced to learn from their past executions and adapt to perform better in the similar situations.

Steiner et al [7] describe a software agent architecture composed of three units: The Communicator, the Head, and the Body (Figure 4). The Communicator is

responsible for sending and receiving messages from other agents in the environment. The Head is responsible for maintaining and employing the rules of operation, the rules that enable the agent to make decisions and respond to changes in the environment. The Body is responsible for carrying out the actual domain specific tasks as per the rules of operation.

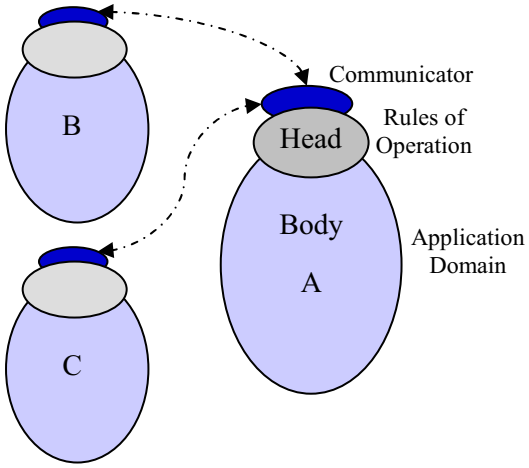

Figure 4. Software agent architecture

An agent-based approach to data warehousing can provide straightforward integration of new data sources into the existing architecture via service provisioning. Software agents can enter and leave the system without disrupting the functionality of the system. As and when new data sources to be integrated into the system are identified, representative wrapper agents can be implemented and added to the system seamlessly. The various data services for the new data source can be easily built into the representative agent. The agent then can advertise the new data source so that the data from this source can be integrated into the repository.

5. Architecture

Our agent-based data warehousing system consists of software agents that provide combinations of the various data-processing services in a heterogeneous, distributed environment, as best-suited for the problem of weather data provisioning and also allow for seamless integration of new data sources and data formats (Figure 5). We use service-descriptions to describe the type and content of the data provided by a source for the agents to publish their services. These services descriptions can then be discovered by other agents in the system. The system has four types of agents, namely Retrieval agents, Repository agents, Translation agents and Application agents.

Retrieval Agents: The retrieval agents provide data extraction, parsing and standardizing services. These lightweight, wrapper style agents capture data as it exists at the data sources, parse the data into Java objects with a pre-determined structure ('QLI_Format') and pass the serialized objects on to their subscribers. The retrieval agents are implemented specific to the data source they are responsible for retrieving data from and are hence light weight in terms of process logic and concentrate only on the process of retrieval. These agents do not maintain any repositories of the data they collect. In the current implementation there is one retrieval agent each for the NCDC, NOAA and FSL websites. The NOAA-agent retrieves data every hour as the NOAA website has transient data. The NCDC and FSL agents retrieve data on demand as the corresponding sites maintain their own repositories and provide URL-based query support.

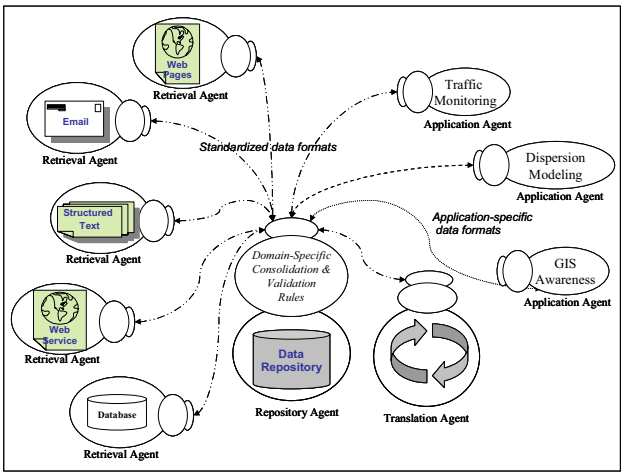

Figure 5. Agent warehousing system architecture

Repository Agents: The repository agents provide a combination of completion, consolidation, persistence, updating, and validation services. Repository agents either subscribe to or send individual requests to the available retrieval agents for their data needs. Repository agents apply domain-specific integration and completion rules to the data. These agents maintain a large repository of data that they subscribe to, (e.g. Weather) in QLI_Format. In our current implementation, the repository agent maintains its data in a MySQL database. Also when application agents request for data in particular formats, these agents look up and employ translation agents for format conversion services. The repository agent uses discovery and integration of services as published by the agents to divide the original data request into service specific requests.

Translation Agents: Translation agents provide data structure conversion services from QLI_Format to other formats. Translation agents apply the information specific conversion rules to carry out their services and are very specialized in particular format conversions. Unlike the retrieval agents, these agents do not have an access to or wrap an data source, but must be given the

data to be converted to desired format. For example, agents that wrap any GIS applications extract a set of weather data attributes from the repository storage format.

Application Agents: Application agents are wrapper-style representatives of various applications in the system and act as information consumers. Application agents look up and either request for or subscribe to repository agents that provide services for the type of data they require.

We have implemented and deployed the described system using our in-house Multi-Agent Development Environment (MADE). MADE enhances JADE (Java Agent Development Environment) [8] with simplified interfaces for agent development and capabilities such as subscription services. JADE is a Java-based middle-layer software framework for agent development and deployment supporting the FIPA standards [10]. The agents communicate via a subset of Agent Communication Language (ACL) message performatives and protocols as implemented in JADE. We also have a particular set of schema of services assuming a common ontology in-place. Agents register and advertise their services, namely the "Retrieval", "Repository" and "Translation" services using the JADE Directory Facilitator (DF). Once registered, the retrieval agents either start retrieving data from their sources or wait for a request for data. The repository agent, after registering its repository service with DF, looks up retrieval services and subscribes to them. The repository agent periodically checks for and subscribes to new retrieval agents that join the system. The application agents look up the weather repository service and subscribe to them for their data requirements. The repository agent looks up and employs translation agent when it needs to supply data in specific format other than the standard format that it stores its data in.

6. Application to Weather Data Warehousing

Information about weather is very important, not only to human users, but also to software applications that require weather data. Such applications include emergency response tools that support scenario planning and building foresight for emergency planners. Many of these tools require different aspects of weather data as input to carry out their processing. Viewing these different types of data on a map of a geographical area helps provide a situational view of the world. In our example, maps also act as ultimate consumers of weather data.

Another application in our example that consumes weather data is a traffic monitoring and forecasting service. Other than analyzing, predicting, and planning for the current traffic purely based upon the time of the day and the road conditions, a traffic monitoring system should take into account current and forecast weather conditions. Warnings such as snow, ice, and high winds can help traffic experts plan ahead in case of traffic conditions due to road accidents. For applications such as the above, weather requirements can range from historical to forecast data.

Weather data is comprised of values of current weather elements such as temperature, pressure, wind speed, wind direction, dew point, precipitation, humidity, etc. Various aspects of weather data are measured by different institutes by means of sensors and other advanced equipment. For example, surface weather data (weather at sea-level) for a given time and location is measured and can be accessed at the National Weather Service website. Weather data is validated within two to three days and can be accessed at the National Climatic Data Center website [24] for a few pre-determined weather stations only.

The system initially is deployed with two retrieval agents, one repository agent, two translation and two application agents. The 'NOAAForecastAgent' retrieves forecast weather data from the NOAA website and the 'NOAAHistoricalAgent' extracts validated historical weather data from the National Climatic Data Center (NCDC) website. The 'WeatherRepositoryAgent' provides complete weather data for a given date and location. The 'MapAgent' and 'TrafficAgent' represent GIS and traffic monitoring and analysis applications, respectively. The 'MapTranslationAgent' provides standard-to-map format conversion and the 'TrafficTranslation' agent provides standard-to-traffic specific format conversions.

NOAAForecastAgent retrieves and parses data from the NOAA forecast service website into standard JAVA objects. The NOAA website is shown in Figure 2. Detailed forecast data is available for the next 60 hours and summarized data exists for the next ten days. NOAAHistoricalAgent retrieves and parses data from the NCDC website on demand. Historical data is available for every hour in the format shown in Figure 3. Historical data for a day only becomes available after 2 to 3 days. Historical data is more detailed than the forecast data in terms of available weather attributes.

The MapAgent essentially represents a GIS application that allows users to view data from the geographical perspective, based upon data such as latitude and longitude. Weather data is typically measured for an area or a point. GIS applications can provide the geographical perspective for weather data by showing their coverage area on the map of a region.

The TrafficAgent represents a traffic monitoring and analysis application that utilizes the wind speed and other daily warnings fields such as heavy rain, snow, sleet, icy

road conditions, etc. of the weather data. As weather data exists in the repository for a region and a time, the TrafficAgent requests and subscribes to current weather data for the geographical regions to which the various roadways belong to.

The MapTranslationAgent extracts certain features from the standard weather data, thereby reducing the set of attributes to those required by the MapAgent. The TrafficTranslationAgent extracts wind speed and other warnings from the standard format weather data.

The WeatherRepositoryAgent stores forecast and historical data into one weather repository as and when it gets updates and new data from the scrapers. The WeatherRepositoryAgent also performs validation, completion, and consolidation of data. It successfully updates the previously existing forecast data when the corresponding historical validated data becomes available.

The NOAAForecastAgent and the NOAAHistoricalAgent register with the JADE directory facilitator service as "WeatherRetrieval" service providers. These agents retrieve data and send update messages with the current results either to their subscribers or to the requesting agent. The WeatherRepositoryAgent registers its 'WeatherRepository' service and performs a DF look-up of 'WeatherRetrieval' service providers. The WeatherRepositoryAgent subscribes to all the resultant agents using the Subscribe messages. The MapAgent and the TrafficAgent, as per their requirements, perform look-up for WeatherRepository services and either subscribe to or send queries to the resultant agent(s) using the Subscribe or QueryRef messages. The MapTranslationAgent and the TrafficTranslationAgent register their 'MapConversion' and 'TrafficConversion' services with the DF, respectively.

The WeatherRepositoryAgent checks to see if the requested weather data exists in the repository. If not, it sends data requests to appropriate agents. It also checks if there is a specific format request and, if so, it accordingly employs the appropriate translator agent. The results received from the translation agents are attached to the InformRef/ QueryRef reply messages to the application agents. For example, for requests from the MapAgent, the WeatherRepositoryAgent looks up and employs any MapConversion service provider.

7. Dynamic Extension of the Weather Data Warehouse

After the work above was completed, we needed to incorporate a new application that required enhanced data from the weather repository. An air quality modeling system models the transport of pollutants in the air following an accidental or intentional release of a biological or chemical substance. CALPUFF [6] is one such air quality modeling tool that is used by the U.S. Environmental Protection Agency (EPA) for modeling long-range transport of the pollutants. CALPUFF computes an approximate plume of the pollutant over a period of time based upon the weather data for that time period. The assessment of the pollutants is done based upon the weather data in a geographical area.

In order to create the plumes, CALPUFF requires upper air data referred to as 'radiosonde' data. Radiosonde data typically consists of values for pressure, temperature, dew point, wind speed and direction, and the height/altitude at which these values are measured. A radiosonde is a scientific instrument equipped with sensors to measure one or several meteorological variables (pressure, temperature, humidity, etc.) The equipment also has a radio transmitter for sending this information to the observing station. Weather stations providing radiosonde data release radiosonde equipment in hot air-balloons up through the atmosphere for measuring weather attributes at various altitudes. The National Weather Service, along with other meteorological service providers, have been recording weather data fields such as temperature, pressure, etc at the altitudes of about 35 kilometers above the weather station surface. In our system, we access radiosonde data for given times and locations from the Forecast Systems Laboratory's (FSL) website [9].

To enable dynamic querying by air quality modeling systems, radiosonde observations must be merged/ fused with the existing surface weather data as depicted in Figure 6. As and when radiosonde data for a time and location becomes available, it is added to the existing surface data for the same time and location.

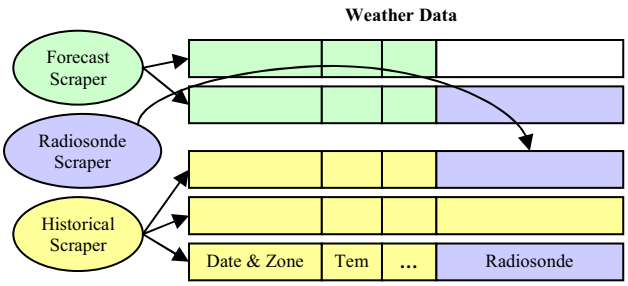

Figure 6. Data fusion for various weather components

To incorporate the new weather data source, we only put efforts into implementing a new retrieval agent; the 'RadiosondeAgent' that periodically retrieves and parses data from the FSL website and passes on the results to its subscribers. The WeatherRepositoryAgent combines the radiosonde data for a particular time and region with the existing weather record for that time and region as mentioned above.

Once the retrieval agent was implemented, there was no requirement to bring down the system to incorporate this new weather data component. Rather, upon launching, the new retrieval agent dynamically joined the system by registering its 'WeatherRetrieval' service with the DF. During its periodic check for new weather retrieval service providers, WeatherRepositoryAgent automatically subscribed to the new retrieval agent, started receiving the new weather data and merging them into its existing repository. Finally to get the newly available data from the WeatherRepositoryAgent to CALPUFF, we merely wrapped the CALPUFF modeling system with a PlumeAgent.

8. Conclusion

This paper has demonstrated how a flexible agent architecture can be used for solving a number of problems associated with collecting data from a number of disparate, independently managed information sources, where the quality, format, and sources of the data may change over time. It allows for dynamically (during runtime) adding agents corresponding to new sources and data formats without altering any other system components. It allows sharing of data in different formats and different subscription models to a wide variety of applications.

A comparison to current approaches indicates that an agent-based approach to data warehousing has many advantages. Our system can seamlessly handle new data sources as opposed to the current statically configured data warehouses. Our architecture also provides middle layer repository agents that not only act as information providers but also support integration and verification of information as per given domain rules. Other approaches handle mostly validated data and adding a validation logic layer at each of the nodes in the network can slow down network flow considerably. Our architecture alleviates such problems too.

We gratefully acknowledge our colleagues Brian Williams, for implementing parts of the described system, and David Cleaver, Apperson Johnson, and Srikanth Kallurkar for their comments and suggestions.

9. Future Work

Our current architecture associates one data source with each retrieval agent. This can be easily extended to have one retrieval agent scrape data from various sources. The important functionality to exhibit here is that if the retrieval agent fails to scrape data from a source, due to changes such as format change, it should continue to scrape data from all its other sources. The data retrieval agents can be enhanced to include a thin verification and consolidation layer to support the above functionality. Another similar direction would be to have discovery style retrieval agents that, given concise information descriptions, can discover the sources. This will also take care of the source failure case, which is not addressed in the current system.

Our current architecture also uses one central repository for the standardized, integrated, and validated data. Implementing a federated network of replicated and/or distributed repositories is a straightforward extension to this architecture.

From the overall architecture point of view, the vision is to have a completely autonomous adaptive network that adapts to the data traffic and determines when and where data should be cached or stored while at the same time optimizing the request-response time.

10. References

[1] A. Levy, "The Information Manifold Approach to Data Integration," *IEEE Intelligent Systems*, 1998, 1312-16.

[2] A. Preece, P.M.D. Gray, and K. Hui, "Supporting Virtual Organisations Through Knowledge Fusion", *Artificial Intelligence for Electronic Commerce: Papers from the AAAI-99 Workshop*, AAAI Press, Menlo Park, CA, 1999.

[3] A. Sheth and J. Larson, "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases", *ACM Computing Surveys*, 1991, 22(3).

[4] AccuWeather at http://www.accuweather.com

[5] C. T. Kwok, D. S. Weld, "Planning to Gather Information," *University of Washington technical report UW-CSE-96-01-04*, 1996.

[6] CALPUFF at http://www.src.com/calpuff/calpuff1.htm

[7] D. Steiner, D. Mahling, and H. Haugeneder, "Human Computer Cooperative Work", *Proceedings of 10th International Workshop on Distributed Artificial Intelligence*, Austin, TX, MCC Tech Re[. ACT-AI-355-90, 1990.

[8] F. Bellifemine, A. Poggi and G. Rimassi, "JADE: A FIPA-Compliant Agent Framework", *Practical Applications of Intelligent Agents and Multi-Agents*, April 1999, pp. 97-108

[9] Forecast Systems Laboratory/ National Climatic Data Center's Radiosonde Data Archive at http://raob.fsl.noaa.gov/Raob_Software.html

[10] Foundation for Intelligent Physical Agents at http://www.fipa.org

[11] G. Wiederhold, "Mediators in the Architecture of Future Information Systems", *IEEE Computer*, 1992, 25(3):38-49.

[12] G. Zhou, R. Hull, R. King, and J.-C. Franchitti, "Data Integration and Warehousing Using H2O," *Data Engineering Bulletin*, 1995, vol. 18, pp. 29-40.

[13] Government of British Columbia's Information Resource Management glossary at http://www.cio.gov.bc.ca/other/daf/IRM_Glossary.htm

[14] I. Muslea, S. Minton, and C. Knoblock, "A Hierarchical Approach to Wrapper Induction", *3rd International Conference on Autonomous Agents*, Seattle WA, 1999, pp. 190-197.

[15] J. Widom, "Research Problems in Data Warehousing", *International Conference on Information and Knowledge*

Management (CIKM'95), ACM Press, Baltimore, Maryland, USA, 1995, pp. 25-30.

[16] K. Decker, K. Sycara, M. Williamson, "Middle-Agents for the Internet", *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, Nagoya, Japan, 1997.

[17] K. Sycara and D. Zeng, "Towards an Intelligent Electronic Secretary", *Proceedings of the CIKM-94 (International Conference on Information and Knowledge Management) Workshop on Intelligent Information Agents*, National Institute of Standards and Technology, Gaithersburg, Maryland, December 1994.

[18] K. Sycara, K. S. Decker, A. Pannu, M. Williamson, and D. Zeng, "Distributed Intelligent Agents", *IEEE Expert*, December 1996, pp. 24-31.

[19] LabCompliance at http://www.labcompliance.com/glossary/c-d-glossary.htm

[20] M. R. Genesereth, A. M. Keller, O. M. Duschka, "Infomaster: An Information Integration System," *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 1997, vol. 26(2).

[21] M. Wooldridge and N. R. Jennings, "Intelligent Agents: Theory and Practice", *The Knowledge Engineering Review*, 1995, 10(2):115-152.

[22] N. Kushmerick, D. Weld, and R. Doorenbos, "Wrapper Induction for Information Extraction", *15th International Joint Conference on Artificial Intelligence (IJCAI)*, Japan, 1997, pp. 729-737.

[23] Napster at http://www.napster.com/

[24] National Climatic Data Center at http://www.ncdc.noaa.gov/oa/ncdc.html

[25] National Oceanographic and Atmospheric Administration at http://www.noaa.gov

[26] National Weather Service at http://www.nws.noaa.gov

[27] O. Etzioni and D. S. Weld, "Intelligent Agents on the Internet: Fact, Fiction, and Forecast*". IEEE Expert*, 1995, 10(4), pp. 44-49.

[28] R. J. Bayardo, W. Bohrer, R. Brice, A. Cichoki, J. Fowler, A. Helal, V. Kashyap, T. Ksiezyk, G. Martin, M. Nodine, M. Rashid, M. Rusinkiewicz, R. Shea, C. Unnikrishnan, A. Unruh, D. Woelk, "Infosleuth: Agent-Based Semantic Integration Of Information In Open And Dynamic Environments", *ACM SIGMOD*, 1997, pp. 195-206.

[29] S. Bergamaschi, G. Cabri, F. Guerra, L. Leonardi, M. Vincini, F. Zambonelli, "Supporting Information Integration With Autonomous Agents", *Fifth International Workshop CIA-2001 on Cooperative Information Agents*, Modena, Italy, September 2001.

[30] S. Bergamaschi, S. Castano, M. Vincini "Semantic Integration of Semistructured and Structured Data Sources", *SIGMOD Record Special Issue on Semantic Interoperability in Global Information*, March 1999, Vol. 28, No. 1.

[31] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom, "The TSIMMIS Project: Integration of Heterogeneous Information Sources", *In Proceedings of the Tenth Anniversary Meeting of the Information Processing Society of Japan*, December 1994.

[32] W. Labio, Y. Zhuge, J. L. Wiener, H. Gupta, H. Garcia-Molina, and J. Widom, "The WHIPS Prototype for Data Warehouse Creation and Maintenance," *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*, Tucson, Arizona, 1997.

[33] W. W. Cohen, "The WHIRL Approach to Data Integration," *IEEE Intelligent Systems*, 1998, vol. 13, pp. 20-24.

[34] Weather.com at http://www.weather.com

[35] Y. Arens, C. Chee, C. Hsu and C. Knoblock, "Retrieving and Integrating Data from Multiple Information Sources", *Journal of Intelligent and Cooperative Information Systems*, 1993, Vol. 2.