



UNIVERSITY OF
CALGARY

Data Integration using Agent based Mediator-Wrapper Architecture

Tutorial Report

For Agent Based Software Engineering (SENG 609.22)

Presented by: George Shi

Course Instructor: Dr. Behrouz H. Far

December 17, 2002

TABLE OF CONTENTS

ABSTRACT	3
1. INTRODUCTION	3
1.1. MEDIATOR-WRAPPER ARCHITECTURE	3
1.2. LITERATURE REVIEWS ON EXISTING DATA INTEGRATION	4
1.3. ISSUES WITH EXISTING DATA INTEGRATION METHODS	8
1.4. AGENT BASED SOFTWARE ENGINEERING	8
1.5. AGENT BASED SOFTWARE SYSTEMS	9
2. AGENT BASED MEDIATOR-WRAPPER ARCHITECTURE	10
3.1. SOFTWARE ARCHITECTURE	10
3.2. THE MEDIATOR AGENT	10
3.3. WRAPPER AGENTS	10
3.4. THE KNOWLEDGE BASE FOR COMMON MODEL	11
3.5. KNOWLEDGE LEVEL COMMUNICATION	11
3. SUMMARY AND CONCLUSIONS	11
4. REFERENCES	11

Data Integration using Agent based Mediator-Wrapper Architecture

George Shi

Dept. of Electrical and Computer Engineering

The University of Calgary

2500 University Dr., NW, Calgary, Alberta, Canada T1N 1N4

gshi@acm.org

ABSTRACT

Organizations have to integrate multiple, distributed data sources and repositories for making their business decisions. These data sources include (*but are not limited to*) databases, object stores, knowledge bases, files systems, digital libraries and legacy systems. Data and/or system integration has become increasing important for enterprise computing. An agent-based mediate-wrapper software architecture is proposed in the course report for integrating multi-source data.

Key words: Agent-based software system, mediator-wrapper architecture, data integration and system integration.

1. INTRODUCTION

1.1. MEDIATOR-WRAPPER ARCHITECTURE

A common problem faced by many organizations today is that of multiple, distributed information sources and repositories including databases, object stores, knowledge bases, files systems, digital libraries, information retrieval systems, ...etc. Decision making process needs information from multiple, heterogeneous, distributed data sources. Therefore, information integration from heterogeneous data sources becomes increasingly important. Data integration of multiple sources requires creating some form of integrated view to allow for distributed querying. The issues for multiple data source integration are summarized [9] as

- The number of data sources may be very high, and makes view integration and conflict resolution a big challenge.
- Adding and dropping a data source should be done with minimal impact on the integrated view.
- Data sources vary in computing capabilities: flat file, spreadsheet, full-featured DBMS (Database Management System).
- Unstructured, semi-structured, structured in nature and provide virtually no information for view integration

To address these problems, the mediator-wrapper architecture was proposed by some researchers [9].

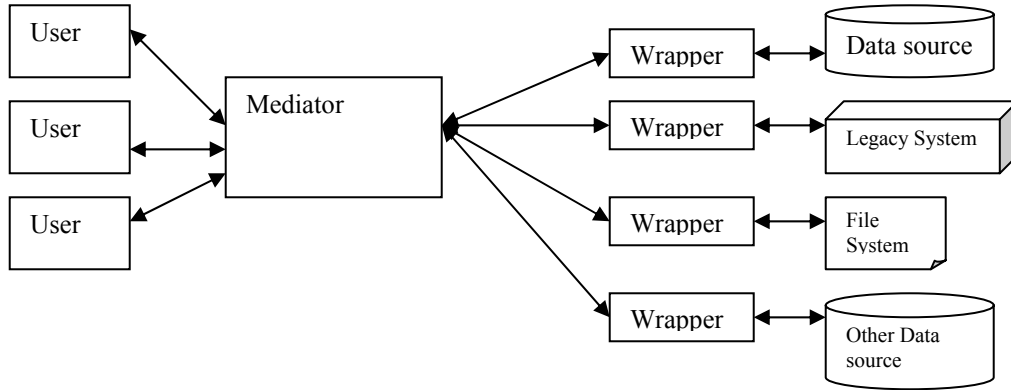


Figure 1. Mediator-Wrapper Architecture [9]

Wrappers are used to provide access to heterogeneous data sources. For each data source, a wrapper exports some information about its source schema, data, and query processing capabilities. A mediator stores the information provided by wrappers in a unified view of all available data with a central data dictionary. It decomposes the user query in smaller queries that can be executed by wrappers, gathers the results from wrappers and creates answers to the user query.

Although implementations of wrappers and mediators vary, the mediator-wrapper architecture has been widely accepted for solving the information integration problem especially for Web base application development.

1.2. LITERATURE REVIEWS ON EXISTING DATA INTEGRATION

In this section, two examples of using Mediator-Wrapper architecture are introduced.

TSIMMIS

TSIMMIS (*The Stanford-IBM Manager of Multiple Information Sources*) [2,4] is a project that developed some tools for facilitating the rapid integration of heterogeneous information sources that may include both structured and semi-structured data. TSIMMIS is based on mediator-wrapper architecture as shown in Figure 2.

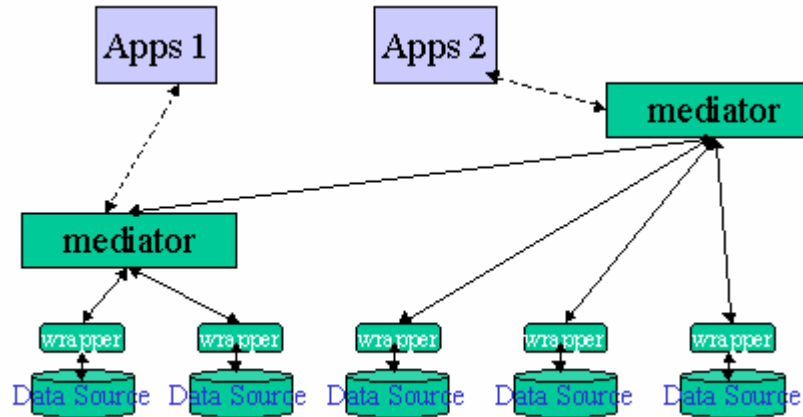


Figure 2. TSMMIS architecture [4.6.7]

For each data source there is a corresponding translator (or wrapper) that logically converts the underlying data objects to a **common information model**. The translator converts queries over information in the common model into requests that the data source can execute, and it also converts the results returned by the data source in to the common model.

TSIMMIS also provides a toolkit to extract properties from some well-structured data source like relational database and translate information into a “common object model” automatically. For those semi-structured or unstructured data sources, wrappers should be constructed on case-by-case base.

TSIMMIS uses OEM (Object Exchange Model)[6] as its common information model. All the data sources have to be modeled in to OEM.

OEM is very simple and powerful model for describing data. Suppose we have a piece of personal data as shown in Figure 3.

```
<person-record, set, {name, address}>
  <name, string, "George">
  <address, set, {road-num, road, city-province, postcode}>
    <road-num, integer, 123>
    <road-name, string, "university Dr.">
    <city-province, string, "Calgary, Alberta">
    <postcode, string, "T2N 1N4">
```

Figure 3. Person data described in OEM

An OEM object consists of a label: person-record, a type: set, and value. The label is just a string represent the name of the object. Type can either be primitive like integer, double or set. If type is primitive, the value field contains the real primitive value like 234 or “city”. If type is “set”, the value field contains a set of

values, which could be sets or primitives. The OEM can be represented as labeled graph like Figure 4.

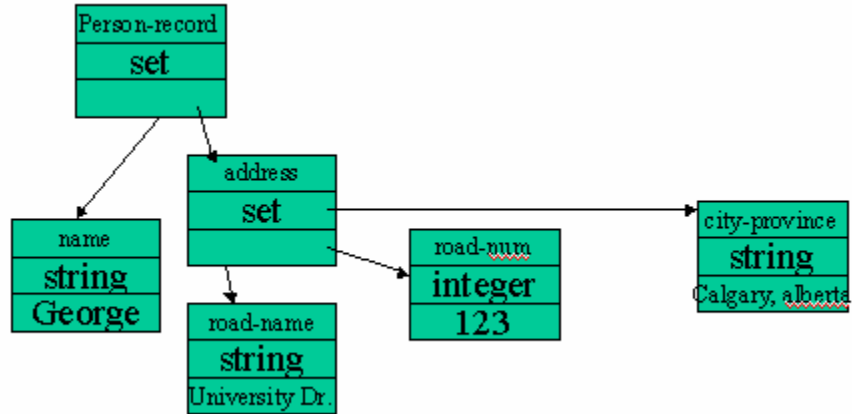


Figure 4. Labeled graph for OEM model

An SQL-like language called OEM-QL is provided for querying data organized based on OEM:

*Select Fetch-Expression
FROM Object
WHERE Condition*

The query results are also OEM objects: $\langle answer, set, \{obj1, obj2, \dots\} \rangle$

GARLIC System

Applying Mediator-Wrapper

Garlic [5, 8] is a middleware system developed on Mediator-Wrapper architecture. It can provide an integrated view of a variety of heterogeneous legacy data sources, without changing how or where data is stored. Wrappers encapsulate the underlying data and mediate between the data sources and Mediator, which is the Garlic Middleware, itself (Figure 5).

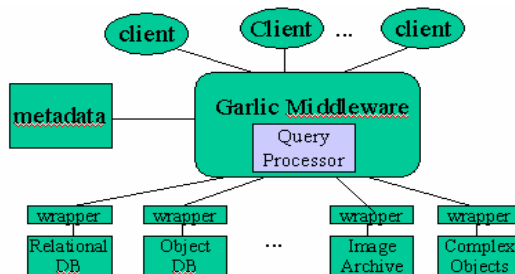


Figure 5. GARLIC Architecture

Wrappers

Wrappers map their data sources to the Garlic middleware model by transforming underlying schema and data. They provide an OO view of the data sources to the Mediator by modeling all collections of similar items in class interfaces. For each interface, there may be more than one implementation, each of which corresponds to some subset of that collection, located in some data sources. The wrappers are objects themselves. Their methods correspond to supported operators. Query planning is done by invoking the wrapper methods and deducing from the return results how much the wrapper can handle.

The Mediator

Mediator in Garlic is implemented as middleware and model heterogeneous data stored in a different source as instances of objects in a unified schema and common interface. Therefore it can leverage the storage and data management facilities provided by legacy systems and provide a unified schema and common interface for new applications without disturbing existing applications

The Garlic middleware merges the schemas of individual data sources into the "global schema" via a wrapper registration step. In this step, wrappers model their own data as "Garlic objects" and provide an "interface" definition that describes the behavior of these objects. Through the "interface" definition, wrappers can rename objects and attributes, change types, change relationships. For example, a relational wrapper might model foreign keys as relationships. The interface is described using the Garlic definition language (GDL), which is a variant of the ODMG ODL [10].

For semi-structured or unstructured data sources, modeling progress would be very much manual. For structured data sources, modeling can be automatic. For instance, a relational wrapper can decide on a common mapping between the relational model and the Object Oriented model (Garlic unified model):

<u>Relational</u>		<u>Object Oriented</u>
<i>Tuple</i>	\Rightarrow	<i>object</i>
<i>Column</i>	\Rightarrow	<i>attribute</i>

Query

Garlic middleware provides a query processor (or called query engine). It optimizes and executes queries over different data sources posed in an object-extended SQL. Its query engine is capable of executing any extended SQL operation against data from any source. In both planning and executing the query, it communicates with "wrapper" for the various data sources involved in the query.

1.3. ISSUES WITH EXISTING DATA INTEGRATION METHODS

- Although Mediator-wrapper is an effective approach for data integration, one of the challenges is to define a common model.
- Since communication between wrappers and the mediator is still at “data” level not “knowledge level”, wrappers have to be developed based on the data source and comply with predefined common model. The more structured the data is, the more automatic the wrapper building process will be.
- Scalability.
Most of existing implementations of Mediator-Wrapper Architecture do not scale well.
- Flexibility
Because of tight coupling and no autonomous, most of existing data integration systems lack of flexibility.
- Adaptability
Most of the existing data integration systems are hard to adapt to changes because of high coupling between wrappers and mediators.

Because of limitations of mediate-wrapper software architecture, we need to look for other advanced implementation paradigms to solve these data integration problems. Agent based software is one of the most interesting solutions.

1.4. AGENT BASED SOFTWARE ENGINEERING

An increasing number of software projects are being revised, restructured and reconstructed to employ the MAS (Multi-Agent System) paradigm to solve problems caused by scalability, complexity and heterogeneity[1,2,3]. Although Agent-based software engineering is far from mature, there are analysis/design methods, research, frameworks, architectures having been proposed by people from different background. From requirement gathering, analysis, design, to implementation, all these proposed technologies have been experimented for developing agent based software.

Research on the theoretical foundations for Agent based software engineering have been focused on: Artificial intelligence, Operational Research and Software Engineering[2]. Technologies such as Ontology, Communication, Uncertainty, etc. have been introduced to solve problems in developing Agent based software systems.

With increasing interests and business potential, Agent based technologies moves from theories to practices. In next session, a few examples of Agent based software systems are briefly introduced.

1.5. AGENT BASED SOFTWARE SYSTEMS

A software control system for Robot [11]

IMA (Intelligent Machine Architecture) is an agent based control software system for the Robot. It allows the concurrent execution of software agents on separate machines while facilitating extensive inter-agent communication. This system was constructed using agent based, parallel and distributed software architecture.

As described in [11], the system was designed from a “complete agent perspective”. An agent within this control system incorporates all aspects of the agent from sensory stimulation in its real environment through its motor actions which is different from the more common approach of separately studying different aspect of sensing, modeling, planning, and action then recombining those results into a model of the agent. Within this system every subsystem is implemented as an Agent. They are loosely coupled which facilitates the parallel processing.

An Agent based software development support system[12]

[12] introduces an agent –based system for capturing and indexing software design meetings. This is a typical example of applying Agent based technologies to software development process itself. In fact, it is meant to be designed as a support tool for software development.

An agent-based software infrastructure called Metaglué is introduced. Metaglué, a multi-agent system (MAS), is the foundation for the development of the project of “Intelligent Room”.

Metaglué supports for synchronous and asynchronous communication among distributed systems. It provides mechanism for resource discovery and management. Reliability is built on the recovery mechanism. Metaglué supports for multi-modal interaction through speech, gesture and graphic user interfaces.

Agent based E-commerce Systems

There are many e-biz or e-com applications with Agent based technologies applied to some extent. For instance, Google applied Ontology in its search engine. Agent cooperation, coordination, competition technologies have been used in online trading and auctions applications.

2. AGENT BASED MEDIATOR-WRAPPER ARCHITECTURE

To solve the problems of data integration using Mediator-Wrapper architecture, the author proposes an Agent based Mediator-Wrapper architecture.

3.1. Software Architecture

Mediator and wrappers are implemented using Agent based technologies. Each agent will have knowledge-ability, learning, autonomy and communication features built in.

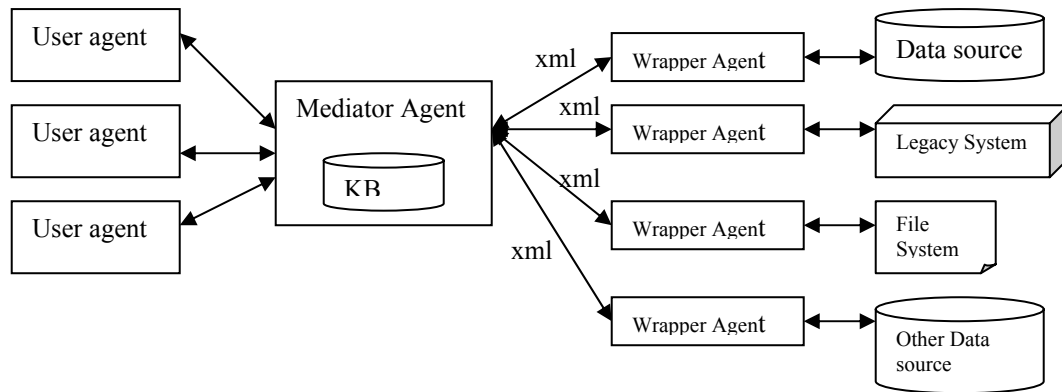


Figure 6 Proposed Agent based mediator wrapper architecture

3.2. The mediator Agent

The mediator Agent acts as a facilitator to coordinate the operations among agents. The common data model is defined in the knowledge base. The mediator agent should have self learning ability to adapt itself for new data sources. Therefore it would provide high scalability.

3.3. Wrapper Agents

Because the integrated system has to exchange data with different sources, each wrapper agent has to provide ability to convert specific scheme to the common scheme stored in the Mediator Agent's knowledge base. On the other hand when the mediator passes request to a specific data source wrapper agent, the wrapper agent has to convert the knowledge query to the specific query that the data source would recognize.

3.4. The knowledge base for Common model

XML is used to describe the data model and data exchanging format between the mediator agent and wrapper agents. Although it is a simple and straightforward data format, we may encounter few drawbacks especially in performance. XML encoding and decoding consumes lots of resource and slows down the application server. The common data model's DOM has to stay in memory.

3.5. Knowledge level communication

Conventional mediator-wrapper architecture implementations use data level communication. The agent based mediator-wrapper architecture will apply knowledge level communication. Ontology will be applied for knowledge sharing and management.

3. SUMMARY AND CONCLUSIONS

By reading publications about Agent based software and Mediator-Wrapper architecture patterns, I found that combination of these two technologies would solve some issues with the data integration. Since the demand for web information integration is increasing, it is worth to further investigate and try a prototyping.

4. REFERENCES

- [1] C. Petrie, *Agent-Based Software Engineering*, <http://nrc.stanford.edu/~petrie/agents/abse/abse.html>
- [2] B. H. Far, *Lecture notes of SENG 609.22: Agent-based Software Engineering*, <http://www.enel.ucalgary.ca/People/far/Lectures/SENG609-22/index.html> Fall, 2002, The University of Calgary, Calgary, Alberta, Canada
- [3] N. R. Jennings, and M. Wooldridge *Agent-Oriented Software Engineering in Handbook of Agent Technology*, (ed. J. Bradshaw) AAAI/MIT Press, 2000.
- [4] TSIMMIS Home Page, <http://www-db.stanford.edu/tsimmis/tsimmis.html>
- [5] M. Roth, P. Schwarz, *Don't Scrap It, Wrap It! A Wrapper Architecture for Legacy Data Source*, Proceedings of 23rd VLDB Conference Athens, Greece, 1997
- [6] Y. Papakonstantinou, H. Garcia-Molina and J. Widom. *Object Exchange Across Heterogeneous Information Sources*. IEEE International Conference on Data

Engineering, pp. 251-260, Taipei, Taiwan, March 1995. <ftp://www-db.stanford.edu/pub/papers/object-exchange-heterogeneous-is.ps>

- [7] 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 IPSJ Conference, pp. 7-18, Tokyo, Japan, October 1994. <ftp://www-db.stanford.edu/pub/papers/tsimmis-overview.ps>
- [8] Garlic Home Page, <http://www.almaden.ibm.com/cs/garlic/>
- [9] M.T. Ozsu, P. Valduriez, *Principles of Distributed Database Systems* 2nd Edition Prentice Hall, 1999
- [10] R. Cattell, et al. *Object Database Standard: ODMG-93 (Release 1.2)*, Morgan Kaufmann Publisher, San Francisco, CA, 1996
- [11] R.A. Peters II, et al. *A Software Agent Based Control System for Human-Robot Interaction*, <http://www.vuse.vanderbilt.edu/~rap2/papers/huro99.pdf>
- [12] T. Hammond, K. Gajos, *An Agent based System for Capturing and Indexing Software Design Meetings*, <http://www.ai.mit.edu/projects/iroom/publications/WAID02.pdf>