Configurable Data Integration Middleware System

Alvaro C. P. Barbosa[1]

Departamento de Informática
Universidade Federal
do Espírito Santo
Av. Fernando Ferrari, s/n,
Vitória, Espírito Santo
29060-900 - Brasil
email: alvaro@inf.ufes.br

Fábio A. M. Porto[1]

Departamento de
Engenharia de Sistemas
Instituto Militar de Engenharia
Praça Gal. Tibúrcio, 80
Rio de Janeiro, RJ
22290-270 - Brasil
email: fporto@ime.eb.br

Rubens Nascimento Melo

Departamento de Informática
Pontifícia Universidade
Católica do Rio de Janeiro
R. Marquês de São Vicente, 225
Rio de Janeiro, RJ
22451-900 - Brasil
email: rubens@inf.puc-rio.br

Abstract

This paper presents a new approach for generating configured and flexible middleware systems for integration of heterogeneous and distributed data. The configuration is based on the selection of an adequate set of database services components and flexibility is achieved by adopting framework techniques. A control model checks the configuration, manages the communication between components and dynamically schedules tasks. The objective is to build tailored integration middleware systems, using a minimum number of components. In addition, the use of framework allows for increased software quality and reduced development effort.

Keywords: *Databases systems, component-based systems, frameworks composition, interoperability, middleware systems.*

1 Introduction

Developing data integration middleware systems is not a simple task due to the complexity of supporting multiple data models, complex query processing strategies and transaction control techniques. The TecBD laboratory at PUC-Rio developed HEROS: a Heterogeneous DataBase Management System (HDBMS) [1] [2] [3] [4] [5] during the nineties. During this period, we faced, quite a few times, the need to extend or re-write its code to adapt to new application requirements. One of these applications is the SINPESQ project for the fishery area conceived by the Brazilian Ministry of Environment. The objective of the SINPESQ project was to integrate fish disembarkation data from all fishing colonies spread through the Brazilian shore, lakes and rivers. From a data integration point of view, that meant integrating data from heterogeneous schemas, database systems and from different naming conventions.

Two aspects of the design of HEROS made its straight adoption by the SINPESQ project difficult. Firstly, its centralized architecture posed a bureaucratic shackle for distant fishing colonies willing to publish their disembarkation data. Secondly, the application is query based and does not require, neither allows, for updates in local data, eliminating the need for the complex and *heavy* heterogeneous transaction management module existent in HEROS.

Motivated by the SINPESQ project we decided to extend our experience in HDBMS systems and investigate new approaches to system's architectures that could produce configurable and flexible systems. In these lines, we initially proposed the ECOHOOD Project (Environment of Configurable Heterogeneous Object-Oriented Databases) [6], whose objective was to build configured DBMS adapted to specific application domains. The idea was to model systems as an Object Oriented framework. The framework technique is a software engineering technology for producing a reusable, "semi-complete" architecture that can be specialized to produce custom applications [7] [8]. The reuse supported by a framework is at a larger granularity than classes, contributing to organize large system development.

Although the adoption of the framework technique had given us a means to provide flexibility into each of the HDBMS modules, it made us face a new problem: the integration of the different frameworks that composed the ECOHOOD system.

This work presents the research developed towards benefiting from the flexibility acquired in constructing software modules for heterogeneous data integration systems using framework technique and proposing a solution for the integration of different framework modules. Our approach is to view each of the HDBMS service as a software

[1] This work was partially developed while the authors were PhD candidates at the Computer Science Department PUC-Rio, TecBD Laboratory.

component with a published interface. A special Control module is added to the system with the responsibility of recognizing the components used in a system configuration and scheduling the execution of component services.

The proposed architecture is implemented in the Configurable Data Integration Middleware System (CoDIMS) which provides an environment for the generation of configured and flexible data integration middleware systems. The environment provides for adequate and lightweight implementation of data integration middleware systems: "what you need, is only what you get" (WYNIOWYG).

The rest of this paper is organised as follows. In Section 2 we comment the background and some related work. Section 3 presents the CoDIMS approach and its environment architecture. Section 4 presents a use case for a CoDIMS configuration. Finally, Section 5 concludes.

2 Background and Related Work

Our research in data integration systems dated back to 1993 where we started the HEROS project. The HEROS system (HEteRogeneous Object System) is an object oriented HDBMS classified as tightly coupled [9]. The HEROS allows for the integration of heterogeneous database systems into a federation so that queries and updates could be submitted transparently to the data location, to local access paths and any existing heterogeneity. In order to cope with some difficulties in adapting HEROS to support new functionality, we started to investigate techniques for building flexible systems as part of the ECOHOOD project. Extending our experience in data integration, we, later, joined other research groups in the ECOBASE Project (Database and Web Technologies for Environment Information Systems) [10], whose objective was to share experiences in the integration of environmental information. One of the aspects explored in the project was to build standards and infra-structure to support the interchange of software modules between the members of the group.

According to [11] and [12], database research groups must explore extensibility and componentization in systems development, in order to generate efficient, flexible, and lightweight systems. In the first half of the nineties, some important projects proposed techniques to produce extensible DBMS. EXODUS [13] and GENESIS [14] were the first projects that proposed a more general approach for DBMS construction through DBMS generators.

Nowadays, the development of software based on component [15] [16] has gained importance as a technique for developing flexible systems. New application requirements can be served by modifying existing components or by adding new ones with compatible interfaces.

In terms of data integration systems, one can find quite a few in literature. Some of them are specific designed to support the creation and maintenance of Web sites, like Araneus [17] and Strudel [18]. Others are based on mediation technique: TSIMMIS [19], DISCO [20] and Le Select [21]. A third category can be classified as database oriented: MIRO-Web [22], Garlic [23], and MOCHA [24]. Generally they are designed having a specific application in mind. Supporting new application requirements may entail quite a strong developing effort. Systems directed towards supporting a wide spectrum of applications are commonly large and heavy which translates into execution inefficiency and complexity in use.

3 The CoDIMS Approach Overview

In this section, we present the CoDIMS approach overview. The CoDIMS is an Object-Oriented environment, based on components and frameworks, designed to provide flexibility and configuration to data integration systems. The configuration is obtained through a Control component that exports the interface of integrated components. The latter are specified in the context of the Control which provides for compatibility checking between the interfaces. In other to flexibilize services implementation, each one is implemented as a framework with hot-spots[7] to be instantiated. Framework component modularity helps to improve software quality by localizing the impact of design and implementation changes, reducing the effort in system maintenance. Through the reuse or adaptation of framework components the environment offers an answer for the generation of heterogeneous data integration systems tailored to specific application requirements.

A data integration middleware system is responsible for providing access to data that is distributed and stored over heterogeneous data sources. Given this general definition, the CoDIMS approach for the development of data integration systems specifies some pre-defined interfaces corresponding to data integration middleware services (DIMS) commonly presented in this type of systems, which includes: Metadata Manager, Query Processing, Transaction Manager, Concurrency Control, Rule Manager and Communication (see Figure 1). For each of these interfaces we provide different components that may be selected into a configured system. In addition, the environment offers a Control component that takes part in any configuration.

Figure 2 shows the class diagram of the CoDIMS environment. Some ready to use components are available for cases where no special behavior is required. New interfaces, corresponding to DIMS not initially previewed, can be added to the environment through their publication in the Control component (see section 3.2) and by providing its implementation.

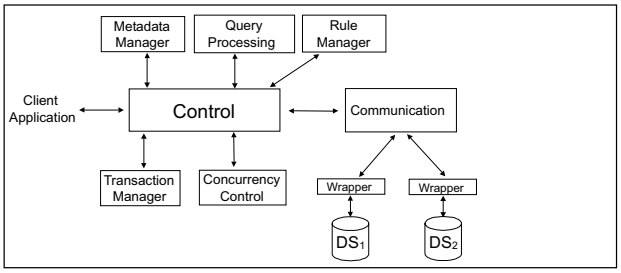

Figure 1 – CoDIMS: architecture overview

The flexibility obtained with these techniques can be summarized as follows:

- DIMS components – allows for the publication of DIMS.
- Framework modules – flexibilizes DIMS behavior.
- Control component – provides for the integration of DIMS into a configured system.
- In the next sub-section we describe each of the pre-defined DIMS offered in th CoDIMS environment.

3.1 System Components Description

The CoDIMS environment comprehends pre-defined DIMS interfaces and components , and the Control module for managing DIMS integration. In this section, we describe the functionality of the pre-defined DIMS:

- The Metadata Manager component is responsible for managing data source and global schema metadata. The environment provides the flexibility of generating a configured system based on a global data model adequate to the application requirements. The access to the global schema follows the Metadata Manager published interface implemented according to the behavior of the selected global data model. Obviously, the adoption of a global data model implicates in special implementation of some of the others DIMS, such as the Query Processing and Concurrency Control. The Metadata Manager DIMS component implements the data integration strategy following the four abstraction levels presented in [9], corresponding to the local data model, exported data model, global data model and external data model. Local data model expresses the data in the data sources. In order to allow the metadata integration, the local data model is transformed into the exported view in the global data model. Finally, the global data model is produced from the integration of a set of exported views.
- The Query Processing component (QPC) is responsible for generating a global query execution plan, for global requests issued by applications, execute them and return the result sets. The QPC DIMS accesses query objects global metadata for query syntactic analysis and optimization. It produces a global query execution plan made of data sources sub-queries and global integration operations. Next, the sub-queries are submitted to the correspondent data sources through the Communication component. Finally, the result of each sub-query is integrated into the final result to be returned to the client application.

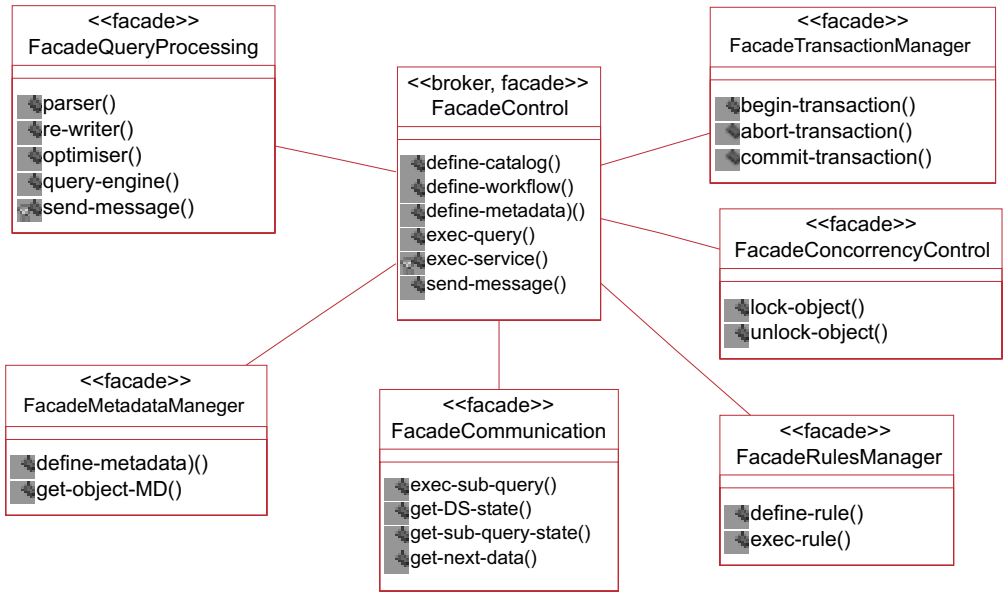

Figure 2 – CoDIMS: Class Diagram

- The Communication component is responsible for the physical communication between the middleware system and each data source. A wrapper translates local sub-queries into specific data source access method calls, as in others mediator systems.
- The Transaction Manager component is responsible for guaranteeing the ACID properties (Atomicity, Consistency, Isolation, and Durability) of transactions, for applications where modifications in the data sources data are permitted.
- The Concurrency Control component provides the mechanisms for implementing the isolation of concurrent transactions.
- The Rule Manager component enriches the system with active behavior as proposed in [26].

The DIMS components are integrated via the Control component which is described in the following sub-section.

3.2 The Control Component

The Control component is the *essence* of the CoDIMS environment. The Control stores, manages, validates, and verifies both Physical and Logical Configuration. Physical configuration corresponds to the selection of DIMS components, their customization according to application requirements, and registration in the catalog. The selection of DIMS components is subject to restrictions. The restrictions are specified as a set of offered and required operations. By matching restrictions between selected DIMS in a configuration, the Control component validates it. The idea of the Logical Configuration is to extract from the system the integration logic modeled by components interaction. The CoDIMS approach achieves a complete adaptability in terms of services to be executed in a configured system, modeling their interaction through a *workflow* [27] of DIMS invocation. Although a pre-defined workflow exists covering the operations over known DIMS, new functionality may be added to the system requiring the specification of a workflow to be evaluated in the event of the new stimulating operation. The adoption of a workflow model brings flexibility in adapting to new query languages, besides providing for detaching of DIMS components. During execution, the Control component automatically responds to client requests by scheduling DIMS services, based on its workflow, and proceeding with DIMS invocation.

The Control component acts as a broker to requests generated between components. The communication between the Control component and the other components is held through their interfaces. The use of a broker diminishes the number of interdependencies between DIMS allowing for increased flexibility and facility in the selection, substitution, modification, and in the relationship between components of the configured system. Differently than CORBA based systems where the control and schedule of services invocation is left to the application, in any CoDIMS generated system, the Control component plays automatically this role.

Figure 3 shows the Control component class model, its facade and its three main modules. In the diagram, some specific notation expresses the facade class as *extensible*, symbolizing its implementation as an OO framework and *static*, meaning that the module is not re-configurable during execution.

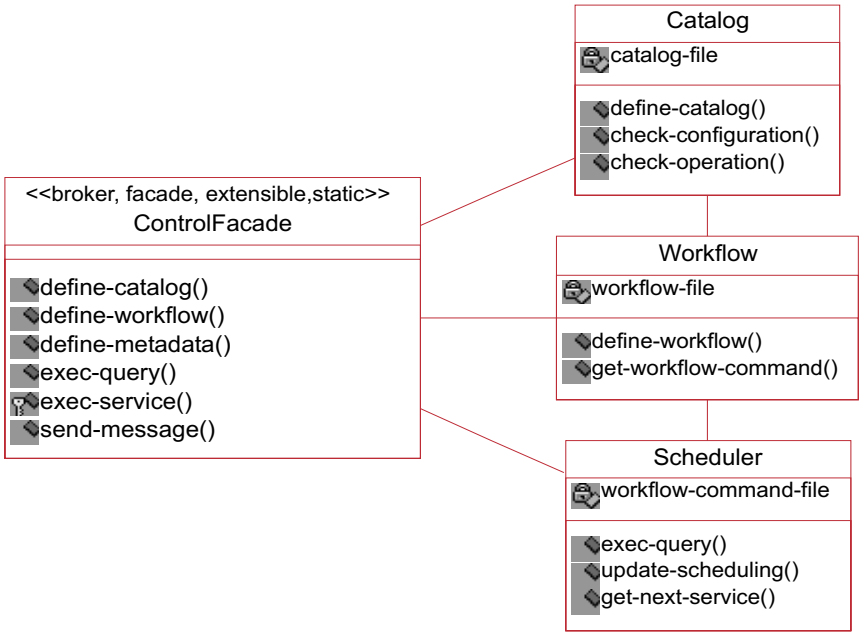

Figure 3: Control Component

The Catalog specifies the physical configuration, which registers each DIMS component present in a configuration, including its name and the offered and requested operations. The Workflow module is responsible for the logical configuration, which register the mapping between DIMS components operations for each global command request. The Scheduler module consults the workflow in order to schedule the respective operations in execution time.

The user of the CoDIMS environment generates a configuration through a process that we present the next subsection.

3.3 – The Configuration Process

The Control, Communication, Query Processing, and Metadata Manager components must exist in all configurations. According to the application requirements, other components may be included. The range of possible configurations of CoDIMS can vary from a midlleware with the functionality of a simple wrapper to a complex HDBMS. In the case where a full HDBMS functionality is required the middleware system could incorporate the Transaction Manager and Concurrency Control components.

The process of generating a specific configured system comprehends the following phases:

a) Design: the system designer selects the necessary DIMS components. In this phase, new DIMS components may be projected or existent ones may need adaptation, customizing semi-complete service implementation;

b) Configuration: the system configurator registers the physical and logical configuration in the Control component. For this, it is necessary to provide two script files to be executed by specific operations: *define-configuration* and *define-workflow;*

c) Load Metadata: the application designer (database administrator) defines local, external, and global metadata through three scripts files to be processed by the *define-metadata* operation. The specific metadata type to be defined is passed by a parameter when this operation is invoked.

During the Configuration step, the *check-configuration* method of the Catalog module in Control component verifies if all the required services are being offered by the component interfaces that participates in the configuration. In a similar way, the *check-operation* method verifies if all operations defined in the workflow are being offered by the specific component. After all these phases, the system is configured and ready for client requests.

4 A Use Case for a CoDIMS Configuration

In this section we illustrate the configuration process within the CoDIMS environment. We refer to the SINPESQ application and one of its integration queries which analyses the predatory fishing activity. The query correlates disembarkation data with species reproduction period. The former is obtained by accessing a data source in the form of an XML file, whereas the latter is stored in a relational database. The disembarkation file, although in XML format, is a flat file with no sub-elements, making its transformation to the relational model, the global data model chosen for the integration system, through a wrapper a rather straight forward one.

To attend this query based application we must configure CoDIMS properly.

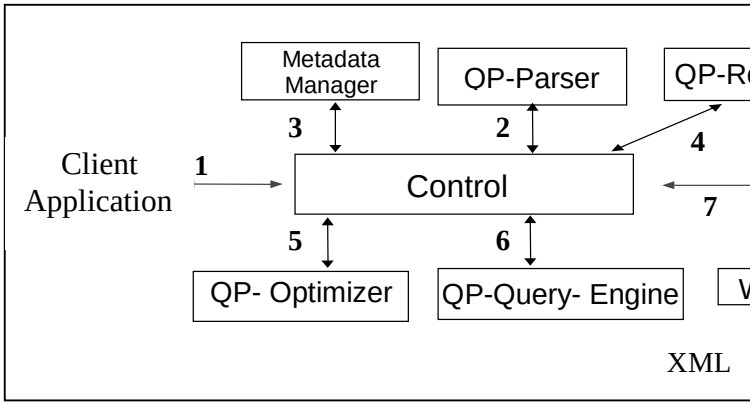

Figure 4: A Configured System Example

Figure 4 presents the configured system components: Control, Metadata Manager, Query Processing, and Communication. The Query Processing component is shown split into four modules: Parser, Re-writer, Optimizer, and Query-Engine. It is important to present the system execution in a better way, as we show later. In this particular configuration, Transaction Manager, Concurrency Control, and Rule Manager components are not required following the query nature of the application.

In order to obtain a configured system for this application, we begin by customizing the selected components. This could be as easy as choosing black-box modules or as hard as coding a specific component from scratch. Once this is done, we must inform to the Control about the components taking part in the configuration, as well as, their order of invocation from an external request. This task, that we call Physical and Logical configuration, is submitted to the Control component, from script files with the needed definitions, and stored in the Control catalog.

In the physical configuration each component presented in the configured system is defined with its both offered and requested operations.

On the left side of figure 5 we present the definition of the Query Processing component with its offered and requested operations, including their parameters. The other components are defined similarly.

We turn now to describe the execution of the global query in the configured system, as illustrated in Figure 4. The Control component receives the query from the client application – number 1 in the Figure 4. The Scheduler module of Control creates an instance of the execution workflow for the SQL *Select* statement and schedules the execution of DIMSs according to the precedence specified in the workflow – numbers 2, 4, 5, 6 in Figure 4. The execution processes as follows: first of all, the Scheduler calls the Parser (2). The Parser executes the lexical and syntactic analysis, consults the global metadata (3) and produces a global query-graph. Next, the Re-writer is called (4) and it groups the information to be supplied for each data source. It transforms the query-graph in an operation tree that contains the sub-queries for submission to each data source based on the global query-graph and on each local export metadata. Then, the Optimizer is called (5) and it generates a global optimized execution plan based on global metadata and export metadata, including the temporary results produced by sub-queries. Finally, The Query-Engine (6) receives and manages the execution of the global optimized execution plan. Each sub-query is sent to specific data source through Communication component (7) and the result of each sub-query is integrated in the final result to be return to the client application.

```
Define Component QueryProceessing
Offered-Operations
  parser(int id, string q-graph, int RC)
  re-writer(int id, string q-graph, string op-tree,  int RC)
  optimizer(int id, string op-tree, string exec-tree, string result, int RC)
Requested-Operations
  meta-data, get-object-MD (int id, string MD-type, string object-name, string
  Communication, exec-subquery (int id, string DS-name, string subquery, s
  Communication, get-next-data (int id, string DS-name, string data, int RC)
End-Component

Define Workflow Select
  Operations
    QueryProcessing (parser);
    QueryProcessing (re-writer);
    QueryProcessing (optimizer);
    QueryProcessing(query-engine);
  End-Operations
```

Figure 5: Physical and Logical Configuration Example

In the logical configuration we provide a workflow that defines DIMS schedule. Considering that the application requires read-only access to data, the workflow maps only the "Select" SQL command. On the right side of figure 5 we present a definition of Select command workflow. Finally, once the system is configured, we need to load the Metadata component with export and global metadata. This is achieved by issuing SQL commands Create Table and Create View, respectively.

The Control module also receives messages coming from invocation of services requested by a component. These messages are not part of the workflow processing and are re-directed to the adequate DIMS component by the Control.

A system prototype for this use case was implemented in the TecBD laboratory at PUC-Rio, using an Oracle DBMS and a XML file. It is written in Java language in order to allow portability. The DIMS components have been devel-

oped and compiled separatedly. The communication between them was implemented using RMI (Remote Method Invocation) which allows for accessing remote components already available.

5 Conclusions and Future Work

Nowadays, there is an increasing demand for accessing and integrating heterogeneous and distributed information such as those available on the Web. Applications built on the top of integrated data vary from read-only to full transactional and ruled-based. Deploying a data integration middleware system capable of coping with such a variety of requirements over a complex heterogeneous environment is a great challenge.

This work presents CoDIMS, a flexible environment for the Configuration of Data Integration Middleware Systems. We develop a component-based architecture integrated through a Control module. A configuration mechanism, based on physical and logical configuration, allows for the selection and integration of a variable set of adequate components. The Control maps client requests into data integration middleware system services and provides for their execution based on workflow precedence. Each individual service component is built as a framework providing for increased flexibility.

The CoDIMS provides for software adaptability through the combination of three techniques: firstly, the selection and integration of adequate components through the physical configuration. Secondly, the customization of each component through the instantiation of an OO framework. And thirdly, the logical configuration that allows for the scheduling of services according to requests of client applications.

The flexibility obtained allows for generating data integration systems supporting specific services and functionality like: an adequate global data model, specific optimization strategies and support for different models and data types. In addition, it allows for the reuse of components or services already available, including remote ones, like: wrappers, parsers, optimizers etc. Moreover, it facilitates the reuse of component substitution and modification in a specific component.

The contribution of this work is an approach for the generation of configurable and flexible middleware system for the integration of heterogeneous and distributed data using components modeled as frameworks.

The CoDIMS is an ongoing project and was the subject of a PhD thesis[28]. At the moment we are complementing the implementation, mainly the Transaction Manager, Concurrency Control, and Rule Manager components. We are also investigating the semantic integration of the components as well as the communication mechanism between them. As future work, we aim to investigate the possibility of dynamically modifying the workflow of mapped middleware services based on the characteristic of a request. As an example, in a global query, if the required data is located in only one data source, the Scheduler module could dynamically re-generate a workflow comprising only of the Parser service and the Query-Engine. The functionality of the Optimizer and that of the Re-writer could be left out, although being defined in the workflow.

References:

[1] Duarte, C.H.C.; Pacitti, E.; Silva, S.D. & Melo, R.N. "HEROS: A Heterogeneous Object-Oriented Database System". *Proceedings of the VIII Brazilian Symposium on Databases*, Paraíba, Brasil, 1993, pp. 383-394. (In Portuguese).

[2] Uchôa, E.M.A.; Lifschitz, S. & Melo, R.N. "HEROS: A Heterogeneous Object-Oriented Database System". *DEXA Conference and Workshop Programme*. Vienna, Austria, 1998.

[3] Uchôa, E.M.A & Melo, R. N.. "HEROS^{fw}: a Framework for Heterogeneous Database Systems Integration". *DEXA Conference and Workshop Programme*. Italy, 1999.

[4] Barbosa, A.C.P. and Melo, R. "Using HDBMS to Access and Dispose Web Information". *Technical Report (PUC-Rio.Inf.MCC 29/99)*, PUC-Rio, Brazil, 29p. (In Portuguese).

[5] Barbosa, A.C.P. and Tanaka, A.K. "Using HDBMS as an Heterogeneous Environmental Information Integrator". *Technical Report (PUC-Rio.Inf.MCC 30/99)*, PUC-Rio, Brazil, 39p. (In Portuguese).

[6] Melo, R.N.; Porto, F.; Lima, F. & Barbosa, A.C.P. "ECOHOOD: Constructing Configured DBMSs based on Frameworks". *Proceedings of the XIII Brazilian Symposium on Databases*, Paraná, Brazil, 1998, pp. 39-51.

[7] Fayad, M.E.; Schmidt, D.C. & Johnson, R.E. "Building Application Frameworks – Object-Oriented Foundations of Frameworks". *John Wiley & Sons, Inc*. 1999.

[8] Barbosa, A.C.P. and Lucena, C.J.P. "Integration of Software Frameworks". *Technical Report (PUC-Rio.Inf.MCC 02/00)*, PUC-Rio, Brazil, 25p. (In Portuguese).

[9] Shet, A.P. & Larson, J.A. "Federated Database Systems for Managing Distributed, Heterogeneous and Autonomous Databases". *ACM Computing Surveys, Vol. 22*, No. 3, September 1990, pp. 183-236.

[10] Tanaka, A.; Valduriez, P. and the Ecobase Project members. "The Ecobase Project: Database and Web Technologies for Environmental Information Systems". *SIGMOD Record, Vol. 30*, No. 3, September 2001.

[11] Bernstein, P., Brodie, M., Ceri, S. et al. "The Asilomar Report on Database Research". *SIGMOD Record*, Vol. 27, No. 4, December, 1998.

[12] Silberschatz, A. & Zdonic, S. "Database Systems – Breaking Out the Box". *SIGMOD Record, Vol. 26*, No. 3, September 1997.

[13] Carey, M.J., DeWitt, D.J., Graefe, G., Haight, D.M., Richardson, J.E., Schuh, D.T., Shekita, E.J., and Vandenberg, S.L. "The EXODUS Extensible DBMS Project: an Overview", in Maier, D., and Zdonik, S. (editors), *Readings on Object-Oriented Database Systems*, Morgan-Kaufmann, 1990.

[14] Batory, D.S., Barnett, J.R., Garza, J.F., Smith, K.P., Tsukuda, K., Twichell, B.C., and Wise, T.E. "GENESIS: An Extensible Database Management System", in Maier, D., and Zdonik, S. (editors), *Readings on Object-Oriented Database Systems*, Morgan-Kaufmann, 1990.

[15] Nierstrasz, O. & Dami, L. "Component-Oriented Software Technology**",** In *Object-Oriented Software Composition*, Chapter 1, edited by Nierstrasz, O & Tsichritzis, D. - Prentice-Hall Inc., 1995.

[16] Pree, W. "Component-Based Software Development - A New Paradigm in Software Engineering?". *Software-Concepts and Tools (18)*, Springer-Verlag, 1997.

[17] Mecca, G.; Atzeni, P.; Masci, A.; Merialdo & P. Sindoni,G**. "**The Araneus Web-Base Management System". *ACM SIGMOD International Conference on Management of Data* , Seattle, USA, May,1998.

[18] Fernandez, M.; Florescu, D.; Kang, Jaewoo et al. "STRUDEL: A Web-site Management System". *ACM SIGMOD International Conference on Management of Data*, Arizona, USA, May 1997.

[19] Molina, H. G.; Hammer, J.; Ireland, K. et al. "Integrating and Accessing Heterogeneous Information Sources in TSIMMIS". *Journal of Intelligent Information Systems*, Vo. 8, No. 2, pp. 177-232, March, 1997.

[20] Tomasic, A.; Raschid, L. & Valduriez, P. "Scaling Access to Heterogeneous Data Source with DISCO". IEEE Transactions on Knowledge and Data Engineering, Vol. 10, No. 5, pp. 808-823, September, 1998.

[21] "LE SELECT: a Middleware System for Publishing Autonomous and Heterogeneous Information Sources". INRIA, English, 1999.

(http://www-caravel.inria.fr/~xhumari/LeSelect/)

[22] Fankhauser, P.; Gardarin, G.; Lopez, M. et al. "Experiences in Federated Databases: From IRO-DB to MIRO-Web". *Proceedings of the 24th VLDB Conference*, USA, 1998.

[23] "The Garlic Project". *http://www.almaden.ibm.com/cs/garlicl* – 16/04/2000.

[24] Martinez, M.R. & Roussopoulos, N. "MOCHA: A Self-Extensible Database Middleware System for Distributed Data Sources". *ACM SIGMOD International Conference on Management of Data*, Dallas, USA, 2000.

[25] Gamma, E.; Helm, R.; Johnson, R. & Vlissides, J. "Design Patterns – Elements of Reusable Object-Oriented Software". Addison-Wesley professional computing series, 1995.

[26] Widom, J., and Ceri, S., editors. Active Database Systems - Triggers and Rules For Advanced Database Processing. Morgan-Kaufmann, 1996.

[27] Jablonski, S. & Bussler, C. "Workflow Management – Modeling Concepts, Architecture and Implementation". International Thomson Computer Press, 1996.

[28] Barbosa, A.C.P.. "Middleware Para Integração de Dados Heterogêneos Baseado em Composição de Frameworks". PhD theses, PUC-Rio, Brazil, may 2001 (In Portuguese).