The Use of Mediation and Ontology Technologies for Software Component Information Retrieval

Regina M. M. Braga
regina@cos.ufrj.br

Computer Science Department
CTU/UFJF
Benjamin Constant, 790,
Juiz de Fora, MG, Brazil,
Zip Code: 36015-400

Marta Mattoso
marta@cos.ufrj.br

COPPE/UFRJ – Computer Science Department
Federal University of Rio de Janeiro
Rio de Janeiro – Brazil - P.O. Box 6851

Cláudia M. L. Werner
werner@cos.ufrj.br

COPPE/UFRJ – Computer Science Department
Federal University of Rio de Janeiro
Rio de Janeiro – Brazil - P.O. Box 68511

Abstract
Component Based Development aims at constructing software through the inter-relationship between pre-existing components. However, these components should be bound to a specific application domain in order to be effectively reused. Reusable domain components and their related documentation are usually stored in a great variety of data sources. Thus, a possible solution for accessing this information is to use a software layer that integrates different component information sources. We present a component information integration data layer, based on mediators. Through mediators, domain ontology acts as a technique/formalism for specifying ontological commitments or agreements between component users and providers, enabling more accurate software component information search.

Keywords
Component Repositories, Domain Engineering, Software Classification and Identification, Component Based Engineering.

1. INTRODUCTION
Component Based Development (CBD) [1] aims at constructing software through the inter-relationship between pre-existing components, thus reducing the complexity, as well as the cost of software development, through the reuse of exhaustively tested components. Building new solutions by combining components should improve quality and support rapid development, leading to a shorter time-to-market. At the same time, nimble adaptation to changing requirements can be achieved by investing only in key changes of a component-based solution, rather than undertaking a major release change. For these reasons, component technology is expected by many to be the cornerstone of software production in the years to come.

According to Jacobson, Griss and Jonsson [1], the effectiveness of component reuse depends on the connectiveness among them and their binding to specific application domains. The connectiveness is one of the most discussed problems in CBD [6, 12]. Approaches that deal with component interfaces[1] (one of premises for connection between components) focus on their capability to provide and request services. Although this interface aspect is important in a CBD approach, other problems arise when trying to connect components. The connectiveness also depends on the execution environment, the heterogeneity of components, the distance between them and the architecture that controls their connections [1, 5, 12]. The architecture that governs the connections between components also depends on the application domain. Therefore, reuse possibilities increase when components are bound to domain concepts. As stated by Krueger [1], while retrieving reusable software components, it is advantageous to use a terminology that is familiar to the domain. This approach diverges from other software component retrieval proposals, such as the Agora System [6] that bases the search only on the component interfaces, covering solely the component connectiveness problem, and the RIG initiative [10] that presents an approach for domain repository integration with minor user transparency and without web access.

Suppose that, in a typical component retrieval scenario, a software developer wants to find software components to use in the construction of an application under development. If he does not know any other specialized service that provides information about components, the natural search space will be the Internet. Now, consider that this developer has no knowledge about the available components. Thus, the following actions are necessary to discover software components that satisfy his needs:

1. To locate information about components that can be stored in distributed repositories. This could be typically done through an Internet search engine that takes as input a few keywords and returns a list of relevant resource descriptions that might be available in these repositories. The success of this task directly depends on the interest of the repository administrators in publicizing their data and the precision of the user while providing the keywords.
2. To determine usability of search results. Due to the complexity in the analysis of component usefulness (i.e., considering the component domain, functionality and connection possibilities based on architecture decisions), the

SSR'01, May 18-20, 2001, Toronto, Ontario, Canada.

[1] An interface of a component can be seen as a component's part that defines its access points. These points allow clients (components themselves) to access services provided by the component [12].

information that is available at the component site is probably not enough.

Thus, a naive Internet approach will not cope with the complexity of software component retrieval. The previous actions require an engine that combines the following three characteristics: (i) Distribution and Heterogeneity – software components can be distributed and use different kinds of storage; (ii) Domain Ontology - to organize component repositories within a domain in order to ease its search; (iii) Software Component Information Evolution – to insert new information (including legacy information).

However, as stated before, current software component retrieval proposals either lack from heterogeneity or domain ontology. On the other hand, many database projects [3,4,7,8,9] are particularly concerned with distribution, heterogeneity (and ontology) found in legacy databases. These projects are known as "multi-database" or Heterogeneous and Distributed Data Base Systems (HDDS) [4]. One solution found in HDDS is the use of mediators[2] [2] combined with ontology [14] to integrate, identify and retrieve related legacy databases. Ontology in this context can be defined as a vocabulary of terms and the relationship between them. For each term, a definition must be created, using an informal description, some concrete examples in the domain, and also a formal specification of the relationships between terms, thus forming a semantic network.

We believe that the HDDS technologies can be adapted to handle software component repositories in the place of legacy databases. Mediators can represent and integrate domain information repositories (distributed and/or heterogeneous). Metadata found in mediators can describe the repositories of components, presenting the domain, their semantics, software architecture and interfaces. Usually a query engine is available in HDDS and therefore ad hoc queries over this metadata can be used to analyze the available components. The organization of mediators with ontology drives the user search along heterogeneous vocabulary.

Therefore, our main objective is to present a software component information retrieval engine named Odyssey Mediation Layer (OML) that combines the connectiveness of components and the domain concept approach. We address both issues through the adoption of mediation [2] and ontology [14] technologies, respectively.

Our approach is motivated by a project that is being conducted in the Legislative House domain. There are several applications that can benefit from reusable information within this domain and from other related ones, such as justice domain, criminal domain, among others. Our users are not specialists in the latter domains, only in the Legislative domain. However, it is important that relevant reusable information from all related domains can be presented to them, particularly when they are not aware of its existence. Most components of legislative process applications can be reused from the legislative domain (e.g., Proposal Creation, Legislature Evaluation, Council Members referee, among others), but sometimes it is worth looking at components from other related domains such as justice domain. Our retrieval engine is able to identify and suggest components from other related domains in the same way as it suggests components from the Legislative domain.

With our retrieval engine, the user search can rely on a controlled vocabulary (ontology) composed of domain terms that are familiar to him. Thus, the search is more focused and executed over relevant available component information repositories. Besides, the usefulness of the retrieved components is supported by the bindings between ontology terms and related components. This binding is accomplished by domain specialists together with domain engineers during a domain engineering process [5, 17] thus enforcing the biding precision. In order to address these issues, the retrieval engine organizes component information repositories within domain ontologies while preserving its original characteristics of distribution and heterogeneity, all in a flexible way.

The main contribution of our proposal is to provide an approach for accessing software components through the use of ontologies and mediators. Our innovative aspect is to provide flexibility, transparency and accuracy in software component information retrieval.

In order to present our approach, the paper is organized as follows: Section 2 discusses the novelties of our proposal with related works; Section 3 details the architecture of Odyssey Mediation Layer; Section 4 shows a component retrieval example; and Section 5 presents our concluding remarks.

2. RELATED WORKS

In a broader approach, several information retrieval systems use semantic brokering with respect to their resources. These systems include SIMS [3], TSIMMIS [4], InfoMaster [7], and Information Manifold [8]. These systems work in the definition of some sort of common vocabulary (similar to an ontology) to define objects in their domain. Individual information sources that contain these objects describe constraints on objects that they can provide, in terms of this common vocabulary. The broker then uses these constraints to determine how to process queries. These projects deal with generic designs for database retrieval. Our approach combines the mediation technology with specific domain ontology [2] to integrate different software components data sources. The main difference between these projects and ours is that our approach is particularly concerned with software components. Thus, we use an ontology, which is specifically constructed for that. This ontology is specified during a Domain Engineering process [5] tailored for this purpose. Hence, the ontology accuracy is more efficient, and consequently the usefulness of the retrieved components.

Another work worth mentioning is the InfoSleuth system[9]. It is a large project, conducted by MCC, which uses an ontology approach to retrieve information from distributed and heterogeneous databases. InfoSleuth can be seen as a framework that can be tailored for a given purpose. One interesting application is the EDEN project in the environmental domain [9]. In this case, some tools of InfoSleuth were customized for this project. Our project adopts a similar approach, where our constructs are specific for software component information retrieval.

The Agora System [6] describes a search engine for retrieving reusable code components, such as JavaBeans and CORBA components. Agora uses an introspection mechanism for

[2] According to Wiederhold [2], mediators are modules that encompass layers of mediation services, connecting bases of heterogeneous and distributed data (producers) to information systems (consumers).

registering code components, through its interface. As a result, this information may not be available at a certain time because the repository is not running, or the information cannot be located. In each of these cases, the interface information cannot be successfully retrieved and indexed, and the component is not registered in the AGORA index database. In our proposal, the use of mediators provides the flexibility to access remote component repositories. There is no need for an index phase, and the mediator is able to capture updates that may occur in a remote repository (i.e., the repository, using the translator[3] services, sends a message with these updates to the mediator). Moreover, the mediator metadata manager has access to all the ontological terms of a given domain, facilitating the identification of the existence of a component, even if its repository is out of service. In this case, the user knows that the component exists and that he can retrieve it latter. Moreover, new information is always associated to domain terms within a given domain ontology, improving its accessibility and reuse.

Ye and Fischer [18] presents an approach that provides an active repository for components. The work emphasizes active delivery of reuse information, helping on the identification of components that developers did not even know that existed. Regarding this last aspect, it is similar to our approach. Our component information retrieval system also provides this functionality too, once it accesses components from other domains based on semantic similarity. The active repository functionality, although not described in this paper, is also part of our work [16]. One aspect that is different in our proposal is the retrieval of distributed information, which is not mentioned in Ye and Fischer's work.

Another important work to mention is the RIG initiative [9], which describes a reuse library interoperability approach. The idea of the asset library interoperability is based on the storage of domain information in several databases. These databases are static and based on a unique global model. The integration requires that information is stored according to this unique model. Therefore, if any reuse database is to be integrated, it has to be translated to the RIG model. Alternatively, the mediation approach creates a new level of abstraction above the database model, allowing the insertion and/or removal of repositories from the mediation structure without the need for updates on the whole structure. Moreover, RIG lacks a more effective search engine that provides searches based on domain concepts and filtering of relevant information, including Internet access, as we do in our work.

3. SPECIALIZING A MEDIATION ARCHITECTURE TO A COMPONENT INFORMATION RETRIEVAL ENGINE

The effectiveness of a software component information retrieval engine is associated to its capacity to handle the distribution and heterogeneity of software components, to organize component repositories within a domain, and to enable software component evolution. These requirements can be accomplished through a software layer that is seen as a particular case of HDDS.

As stated before, mediators are modules that encompass layers of mediation services, connecting bases of heterogeneous and distributed data (producers) to information systems (consumers). Hence, in order to be really useful in software component information search this solution has to be tailored to component information retrieval, considering the component domain, its semantics, architecture, and interfaces.

In a mediation architecture, as new sources of information are aggregated to the mediation structure, the amount of information to be modeled increases, frequently generating inconsistencies, ambiguities, and conflicts in the represented information. One way to deal with this problem is to partition the consumers and producers' models and the structure of mediation by domain. The description of these models, partitioned by domains, forms the so-called domain ontology [14].

In the context of component information retrieval, the use of mediators allows the information access to be carried independently of the format and the operational platform where it is stored. Therefore, the structure of a retrieval engine as a whole can be flexible, since existing component data sources can be added to the architecture in an easy way, with no need to convert from the original information format (format form of data/information source) to the format used by the reuse environment. Another interesting feature of mediators within a component information retrieval engine is that reusable information is naturally organized by domain (Figure 1), which facilitates the search for domain concepts, since specific domain data is accessed in a search. Moreover, the use of mediators allows the aggregation of information already stored in legacy databases, without the need to transform the original database format.

In order to help on the correct choice of mediators for a given domain, the mediator layer provides a specific ontology for each domain. Therefore, this ontology must be specified by domain specialists, facilitating the search for specific components, since the ontology definition is directly connected to software components within the domain.

The use of this layer in the Legislative Domain is particularly interesting, since in Brazil as in other countries [15], there are some legislative houses that are more up to date with software technology than others. The former represents a reference source of software components to several Legislative Houses. Without this kind of layer, there exist some barriers for reusing components among these houses, such as the distance, scarce financial resources, and semantic conflicts among components (a common component functionality can be identified differently in each legislative house).

Figure 1 presents an example of a mediation layer configuration for this specific application domain. Several mediators are presented as sub-domains, such as State Legislative (SL) and Municipal Legislative (ML) domains. The SL Mediator is aggregated (P1) to ML, generating a more generic mediator that combines the two domains. The latter can be used in cases where information concerning the two domains is necessary. Each mediator is connected to the related domain data sources that contain reuse component information. The Justice Domain Mediator may be accessed in cases where the user wants components related to the Justice domain.

In order to provide an architecture that is able to handle the requirements of component information search and retrieval, we specified and implemented OML (Odyssey Mediation Layer) was specified and implemented, based on mediation and ontology technologies. OML is derived from a HDDS mediator, the HIMPAR architecture [12], adding to it more precision and semantics, using ontologies tailored to software component information retrieval.

[3] A translator in this context is the same as a wrapper or adapter.

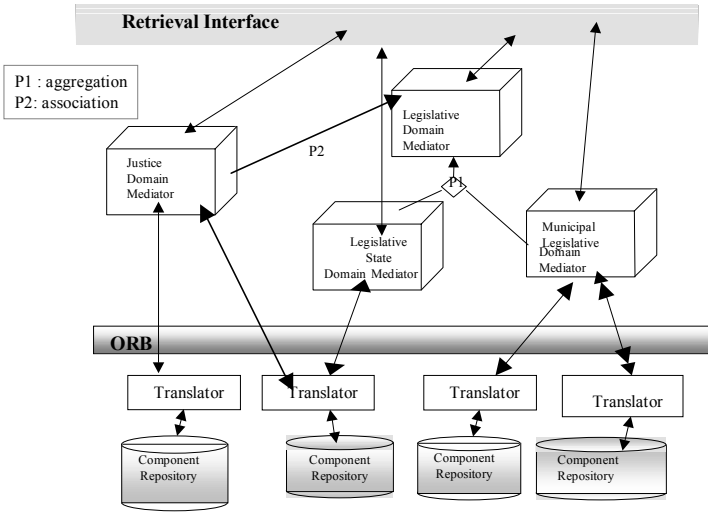

Figure 1 - An example of a mediation layer for the Legislative domain

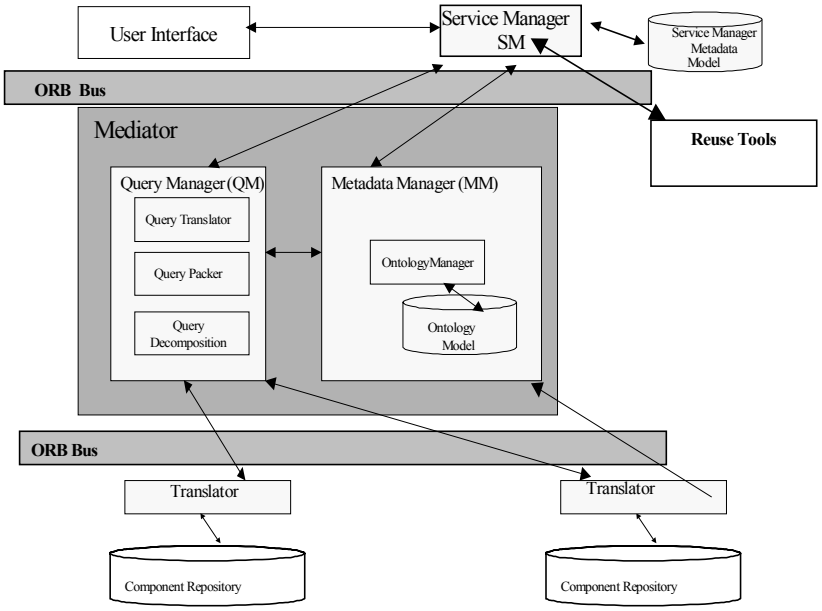

Figure 2- Architecture of the Odyssey Mediation Layer

The OML engine is part of a reuse environment, named Odyssey, that deals with component based development of applications within a given domain. Specifically, OML is part of a system of agents that helps users in their search for reusable components [16]. It uses intelligent mechanisms such as learning techniques and user preferences in order to present in advance components that the user does not even know that exist. The agent system infers this information and presents the components to the user. It is important to notice that although OML was built in the context of the Odyssey project, it can be used standalone or integrated to other tools, since OML offers a user interface (as seen in figures 3 through 8) and a CORBA IDL interface for its communication with other tools.

Figure 2 presents OML, which comprises four levels: Interface, Mediation Layer, ORB bus, and Translators. The Interface level is implemented by the Service Manager (SM), which stores metadata about available mediators, and is capable of creating ontological bindings between related ontologies in order to query several mediation layers. Also, SM is responsible for the creation and modification of mediators. The Mediation Layer provides the management of each mediator through the Metadata Manager

(MM), and provides access to mediators through the Query Manager. At the ORB level, communication between the mediation layer and translators is established through CORBA standard services. Finally, the Translator level provides one translator for each component repository in such a way that it can participate in the Mediation Layer integration model.

3.1 Service Manager

The Service Manager (SM) stores metadata about mediators, translators and data sources availability, and deals with ontological commitments between related mediators (domains).

Schema 1 presents an overview of SM Metadata in ODL notation, and Schema 2 provides the IDL interface to access mediators through the CORBA bus.

```
class Object_Himpar
(  extent Himpares)
{
   attribute string Name;
}

class Mediator extends Object_Himpar
(  extent Mediadores)
{
   relationship list<Container> AssociatedDataSources
    inverse DataSource::Medis;
   attribute string Description;
   attribute string KeyWords;
   relationship list<Mediador> Super
    inverse Mediador::*;
   relationship list<Mediador> Spec
    inverse Mediador::*;
   relationship list<Mediador> Assoc
    inverse Mediador::*;
   attribute string BaseName;
   relationship list<OntologyTerm> TermRel
    inverse OntologyTerm::MediadorRel;
   attribute String password;
}


class Wrapper extends Object_Himpar
(  extent Wrappers)
{
   attribute string description;
   attribute string type;
   relationship list<DataSource> Repositories
    inverse DataSource::Trad;
}

class DataSource extends Object_Himpar
(  extent DataSources)
{
   attribute string owner;
   relationship list<Mapping> Structure
    inverse Mapping::Cont;
   attribute string AbstractionLevel;
   relationship Wrapper Trad
    inverse Wrapper::Repositories;
   relationship list<Mediator> Medis.
    inverse Mediator:: AssociatedDataSources;
   attribute String password;
}

class Mapping
{
   attribute string DataSourceName;
   attribute string map;
   relationship DataSource Cont
    inverse DataSource::Structure;
}

class OntologyTerm
(  extent Terms)
{
   attribute string Name;
   relationship list<OntologyTerm> Synonym
    inverse OntologyTerm::*;
   relationship list< OntologyTerm > Hipernym
    inverse OntologyTerm::*;
   relationship list< OntologyTerm > Hiponym
    inverse OntologyTerm::*;
   relationship Mediator MediadorRel
    inverse Mediator: TermRel;
}

class Component
(  extent Components)
{
   attribute string type;
}
```

Schema 1 – Metadata of Service Manager

The ontological commitments between related mediators are all done at SM level. SM provides the necessary metadata for this, using the Mediator, DataSource, Mapping, OntologyTerm and Component classes described in Schema2. The decision to concentrate all ontological commitments at SM level was mainly based on the SM available information. It knows the availability of all OML components and thus is able to indicate which domain the user could access. In order to use a given mediator within OML, the administrator has to register the mediator, its related data sources, and translators used by these data sources into SM. Figures 3 and 4 present examples of the interfaces for doing this.

```
module Mediator
{
interface Access {
struct OntologyTerm {
                    string Name;
                    string Description;
                    };
struct object {
                    string type;
                    string definition;
                    };
typedef sequence<Ontology> ListOntology;
typedef sequence<object> ListObjects;

// Functions for the management of bases
                    string open_base (in string basename);
                    void  close_base (in string basename);

// Functions for the management of ontologies

                    ListOntology retrieve_Ontology (in string
mediator-name);

// Functions for retrieve components
            ListObjects queryMediator(in string query);
                                        };
                };
```

Schema 2 – SM IDL interface to access mediators

Figure 3 shows some information about the Municipal Legislative Mediator within SM. Some basic metadata are: i) the mediator name, ii) the executable file that has to be loaded (if it is not already loaded) by ORB, in order to respond to some request to this specific mediator, iii) the keywords related to the mediator

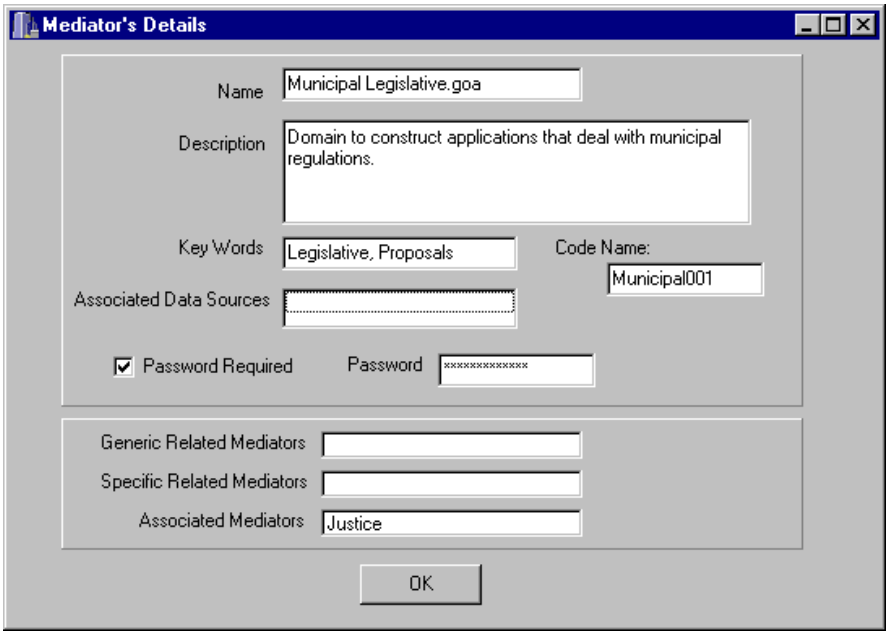

Figure 3 – A Mediator Registration Example

(this information provides a fast and limited knowledge about the contents of the mediator), iv) the password required by the mediator to attend the request (if necessary), among others. Figure 4 presents a data source registration, associating a specific data source, File System2, to the Municipal Legislative Mediator. We may also register the available types of components in order to know to which phase of the application development the component belongs (analysis, architectural or implementation – see Figure 8).

One important characteristic of OML is to use domain ontology to search for domain terms and its ontological relationship, within or among various domains at different levels of abstraction. Thus, SM has to capture the ontological model[4] of each mediator and associate terms among them. For capturing each ontological model, SM uses the ORB bus, through IDL retrieve_Ontology() interface method (Schema 2), to access the specific mediator, retrieving its ontological terms. Therefore, the ontological model provides the main structure for dealing with domain ontology relationships. Relationships involve semantic links such as Hypernyms, Hyponyms, and Synonyms. A Synonym link associates ontological terms in several domains that represent synonyms for a particular ontological term. Hypernyms and Hyponyms links relate ontological terms from various domains that can be either more general or more specific than the current one. Thus, it is possible to associate ontological terms from multiple domains, providing accessibility for domain information. Figure 5 presents the interface for the association of ontological terms.

In a query formulation, SM accesses and retrieves all related mediators, searching for component information that fulfills the query semantics. The ontological information about requested domains is transferred by the Broker (ORB) to the proper domain (mediator), using the method queryMediator (in string query), and correct ontological domain terms.

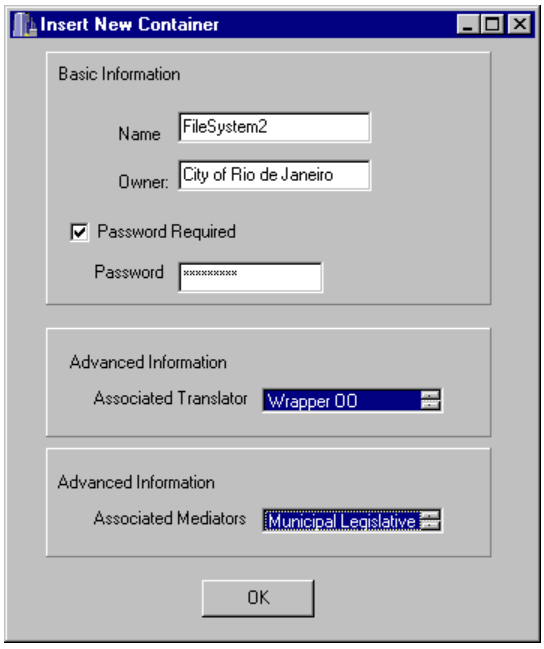

Figure 4 – A data source registration example

In the example shown, SM will query all mediators that are related to the Municipal Legislative mediator. Thus, SM will transmit the query to the Legislative Domain Mediator and Justice Domain Mediator (see Figure 2). The retrieved components and their corresponding match levels are shown, if each retrieved component exactly matches the query or if the component

[4] Each ontological term is specified as a domain term and a detailed description of it, and relationships with other ontological terms, at different levels of abstraction, are created (see section 3.2)

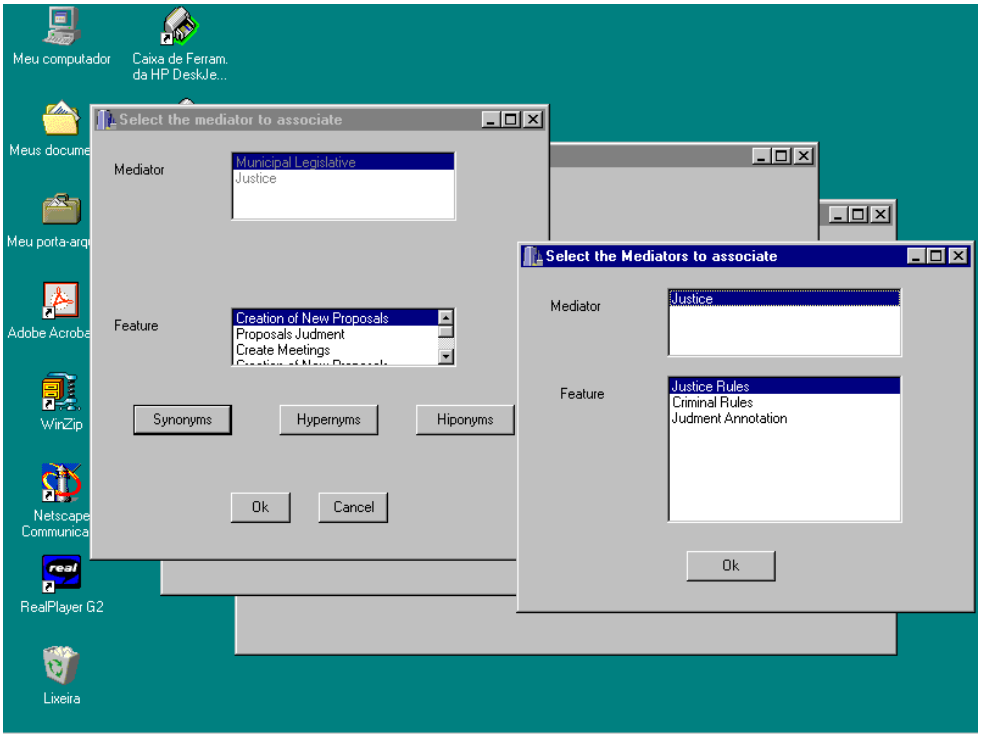

Figure 5 – Multiple Ontology Association

partially attends the request. OML registers this information and presents how well the component attends the request (total or some percentage – in the latter case). In section 4, we present a more concrete example of this kind of query.

3.2 Mediator Manager

Each mediator has its own metadata. This metadata represents the ontological model of the domain. The Mediator manager (MM) also stores relationships among the ontological terms and components stored in data sources. This metadata provides the capability to retrieve components related to this mediator (domain). In order to provide this feature, MM also stores the ontology metadata related to this domain. For each ontological domain term, it is necessary to register its name, its type (in this specific case a term that represents a functionality in the domain), its importance within the domain and the related domain terms. These terms permit the expansion of the query range within the domain, i.e., if there is no component information on data sources related to this specific ontological term, OML could query related ontological terms in the same domain. Of course, this "shift" must be reported to the user.

In order to relate each ontological term with its counterparts in data sources, MM retrieves related information of data sources from SM, using the ORB bus. The retrieved information is used by MM to locate and retrieve software components from data sources. Thus, we can associate each ontological term with its related components, with the help of a specific translator.

4. A RETRIEVAL EXAMPLE USING OML

Consider a user who is developing an application to handle new legislative proposals, among other characteristics, in the legislative domain. He wants to know if he can use pre-existing software components in his application. Thus, he can use the OML user interface to know about the availability of this kind of components, and to retrieve some candidates.

In our example (Figure 6), data source 2 has a binary software component called "New Subject", and data source 1 has a Java package (set of related classes) named "Proposal Creation". Both data sources were mapped into the Legislative Municipal Domain Mediator. Therefore, the Justice Domain mediator has an ontology term named Justice Code that is mapped to a component named Code Database.

These components are made available to the Legislative Municipal Domain through the ontological term in the mediator called "Creation of New Proposals", and are mapped to the above components in data sources, i.e., data sources 1 and 2.

During the creation of new proposals within a Municipal Legislative House, there are some cases when it is necessary to consult justice database rules. This justice database can impose some restrictions on a new proposal creation.

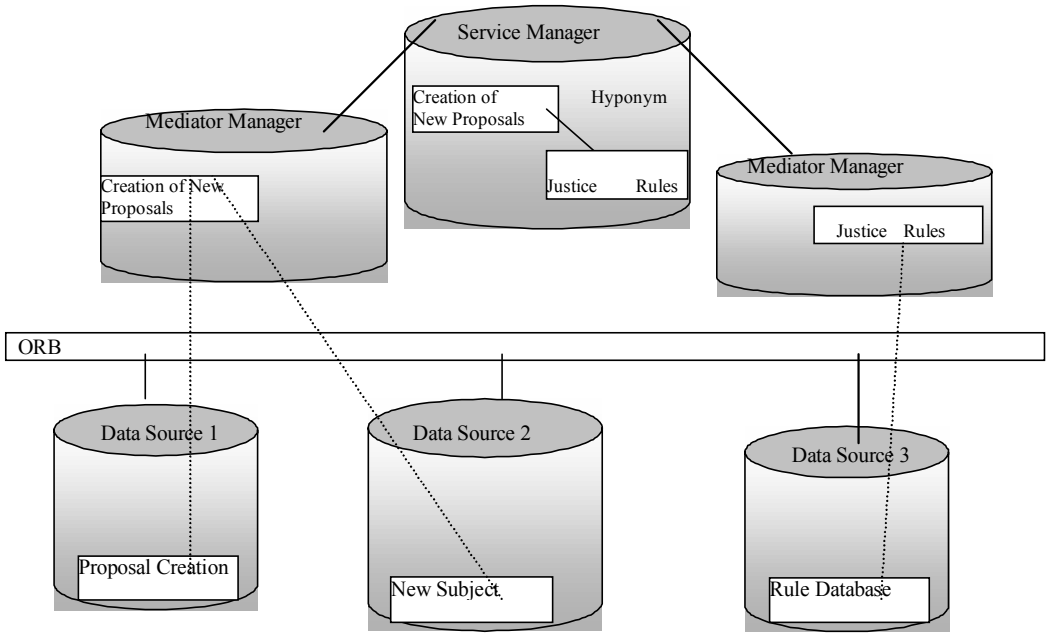

Figure 6 – Retrieval Schema Example

When the Municipal Legislative Mediator was registered, the SM administrator associated this mediator with the Justice Mediator. This Justice Mediator provides software components used for the development of applications in the Justice domain. Thus, when our user accesses the SM interface in order to retrieve components related to the creation of new proposals, he can choose to access information from all related mediators, i.e., generic mediators, specific mediators, associated mediators or all of them. Suppose our user decides to retrieve information from the Legislative Mediator and associated mediators, then he will access components from the Legislative Mediator and Justice Mediator (see Figure 2). The formulation of the query (Figure 7), selecting the type of component to be retrieved (components belong to analysis, architectural, codification or all phases of development), and the result of this query is presented in Figure 8. Note that for each component, a description of the retrieval is presented (in

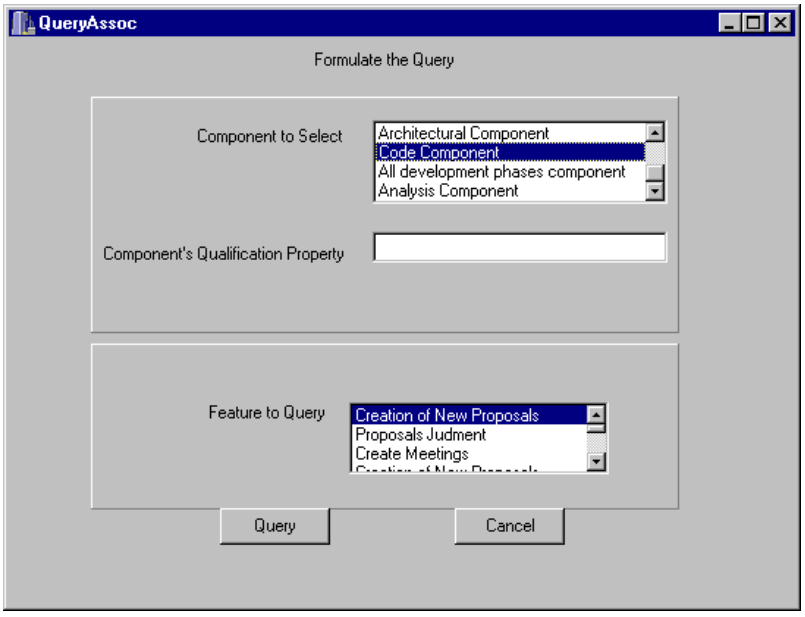

Figure 7 – Query Formulation

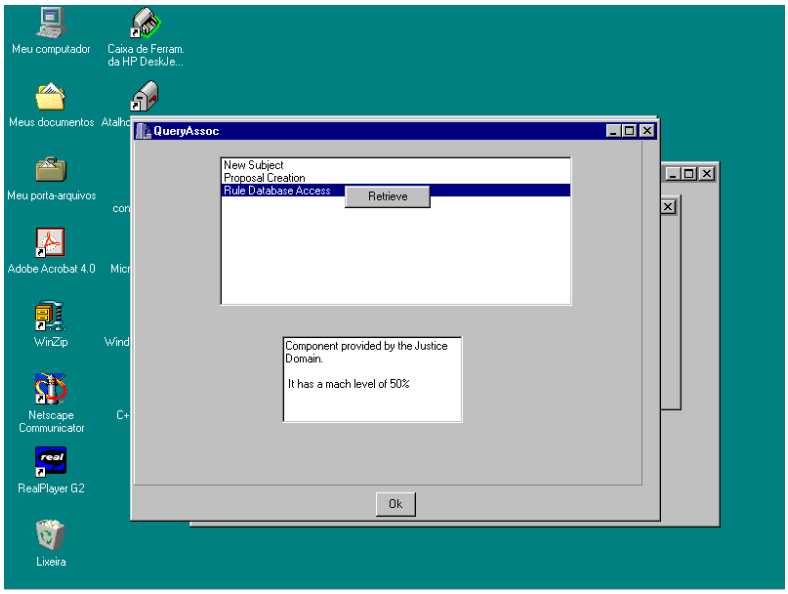

Figure 8 – Example of component information retrieval in OML

Figure 8, the description presented is related to the Rule Database Access component) and the user can select one or more components to retrieve.

Through the mediation structure, OML users can search for components in a transparent and uniform way. In the above example, users of OML do not have to know where components are stored. Moreover, users do not have to query all component repositories, using each specific repository query language format (when a query language exists) to find where the needed components are stored. They do not even have do know how to access data sources.

The complexity for dealing with these heterogeneous repositories is treated by OML. Without this layer, users would have to handle these repositories individually, increasing the complexity of the query and access. By using mediators, users can query specifically the mediation metadata, using one single model. The mappings between mediator metadata and translators redirect and decompose the query to data sources 1, 2, and 3. Also, the identification of components of the same domain that are in different repositories can be detected at the time of their registration in the mediation layer. Afterwards this is all transparent to users.

5. CONCLUSIONS

This work addresses the interoperability problem between component information repositories. An integration layer was developed to help searching and identifying suitable reuse components. This layer is based on mediators and ontologies to provide the binding of different components to their domain concepts. To assist the identification of related components and their appropriate domain organization, each mediator encloses one domain ontology and provides the mapping to their respective repository of components.

Mediators provide a uniform view of the available components organized in domain taxonomy. Domain ontologies are used to help searching for reusable components information through the representation of domain semantic concepts. Therefore, this mediation layer promotes domain information integration and provides mechanisms to translate component requests across ontologies. The important aspect of our proposal is the use of domain ontologies, for reusable component retrieval, in a concrete situation, allowing users to express component requests at a higher level of abstraction when compared to keyword based access or component interface based access used in other proposals. Without OML, users would have to access directly various repositories, dealing with specific characteristics of each repository. Therefore, the main contribution of this paper is to show the potential of the technology of mediators, together with ontology models, for dealing with components repositories complexities, and organizing the manipulation of different components within a domain ontology. Although, the mediation technology is quite popular within HDDS, its adaptation using domain ontologies for component information retrieval is innovative.

OML is an operational interoperability architecture based on the use of mediators, translators, and a CORBA communication protocol, which is responsible for the connection among translators and mediators in a distributed and heterogeneous environment. It was constructed using the C++ language together with the Visibroker ORB for C++. Currently, OML is being extended in order to publish and search for components on the Internet [19], based on XML standard.

References

[1] Jacobson, I.; Griss, M.; Jonsson, P. : “Software Reuse: Architecture, Process and Organization for Business Success;” Addison Wesley Longman, May 1997.

[2] Wiederhold, Gio; Jannink, Jan: "Composing Diverse Ontologies;" 8th Working Conference on Database Semantics (DS-8), Rotorua, New Zealand (DS-8) January 1999 (Final version to be published by IFIP/Kluwer/Chapman&Hall).

[3] Arens Y., Knoblock C.A., and Shen W.: Query reformulation for dynamic information integration. Journal of Intelligent Information Systems, 6(2):99–130, 1996.

[4] Molina, Garcia and et.al. :The TSIMMIS approach to mediation: Data models and languages. Journal of Intelligent Information System, 8(2), 1997.

[5] Braga, R.; Mattoso, M.; Werner, C.: "The Use of Mediators for Component Retrieval in a Reuse Environment," In: Proc. Technology of Object-Oriented Languages and Systems Conference (TOOLS-30 USA'99), IEEE CS Press, Santa Barbara, pp.542-546, August 1999.

[6] Seacord, R.; Hissan, S.; Wallnau, K,: "Agora: A Search Engine for Software Components," Technical Report CMU/SEI-98-TR-011, August 1998.

[7] Genesereth M.R., Keller A., and Duschka O.M.: Infomaster: An Information Integration System. In *SIGMOD RECORD, Proceedings of the 97 ACM SIGMOD International Conference on Management of Dat*a, pp. 539–542, Tucson-Arizona, 1997.

[8] Levy, Alon Y., Rajaraman, Anand, and Ordille, Joann J.: Querying heterogeneous information sources using source descriptions. In Proceedings of the 22nd VLDB Conference, pp. 251–262, Mumbai (Bombay), India, 1996.

[9] Fowler, Jerry, Perry, Brad, Nodine, Marian, and Bargmeyer, Bruce: Agent-Based Semantic Interoperability in InfoSleuth , SIGMOD Record 28(1): pp. 60-67, 1999.

[10] RIG; "Reusable Library Interoperability Group" at http://www.asset.com/rig/, 1996.

[11] Pires, P.; Mattoso, M.: "A CORBA based architecture for heterogeneous information source interoperability;" Proceedings of Technology of Object-Oriented Languages and Systems - TOOLS'25, IEEE CS Press, pp.33-49, November 1997.

[12] Szyperski, C.: Component Software: Beyond Object Oriented Programming, Addison Wesley, 1998

[13] Ram, S.: "Guest Editor's Introduction: Heterogeneous Distributed Database Systems,"; IEEE Computer, Vol. 24 No.12, December 1991.

[14] Nieto, E. M.: *OBSERVER: An Aproach for Query Processing in Global Information Systems based on Interoperation across Pre-existing Ontologies*, Doctoral Thesis, Universidade de Zaragoza, November 1998.

[15] Weinstein, P. C.: *Ontology-based Metadata: Transforming the MARC Legacy*, Proceedings of the 1998 ACM 7th Internacional Conference on Information and Knowledge Management, pp. 52-59, 1998.

[16] Braga, R.; Mattoso, M.; Werner, C.: "Using Ontologies for Domain Information Retrieval," in DEXA 2000 DomE Workshop, pp.100-104, September 2000.

[17] Braga, R.; Werner, C.; Mattoso, M.: "Odyssey: A Reuse Environment based on Domain Models"; In: Proceedings of IEEE Symposium on Application-Specific Systems and Software Engineering Technology(ASSET'99), IEEE CS Press, Richardson, Texas, pp.50-57, March 1999.

[18] Ye, Y.; Fischer, G.: "Promoting Reuse with Active Reuse Repository Systems," IEEE ICSR 2000, Vienna, pp.302-317, June 2000

[19] Pinheiro, R.; Costa, M.; Braga, R.; Mattoso, M; Werner, C.; "Software Components Reuse Through Web Search and Retrieval", *Proceedings of the International Workshop on Information Integration on the Web - Technologies and Applications,* Rio de Janeiro, Brazil, 2001 (to appear).

[5] Each ontological term is specified as a domain term and a detailed description of it, and relationships with other ontological terms, at different levels of abstraction, are created (see section 3.2)