Abstract

CBD (Component Based Development) aims at constructing software through the inter-relationship between preexisting components. However, these components must be connected to a specific domain of application in order to reuse components effectively. In general, reusable components are stored in a great variety of data sources. Thus, a possible solution for accessing domain information is to use a software layer that permits the integration of different component sources. In this paper, we present the use of the HIMPAR integration data layer, based on the use of mediators, which was modified to support a reuse environment. Through mediators, domain ontologies are used as a tool/mechanism for specifying the ontological commitments or agreements between component users and providers.

1. Introduction

Component Based Development (CBD) [1] aims at constructing software through the inter-relationship between preexisting components, thus reducing the complexity, as well as costs of software development, through the reuse of exhaustively tested components. According to Jacobson, Griss and Jonsson [1], the effectiveness of component reuse depends on the connectiveness among them and the specific application domain. Therefore, reuse possibilities increase when components are binded to domain concepts. We propose the search for reusable components at the level of its relevant semantic concepts.

Apart from the component integration difficulty, a reuse environment has to deal with interoperability issues between component repositories. Reusable domain components (in all abstractions levels) can be stored in a great variety of data sources, using the most different data models, access mechanisms and platforms. Moreover, domain components can be geographically distant, incurring in complex manipulations.

Therefore, the effectiveness of a reuse environment is associated to its capacity to handle the following three characteristics: (i) distribution and heterogeneity- domain information can be distributed [2] and uses different kinds of storage; (ii) Domain Ontology- to organize component repositories; (iii) Domain information evolution – to insert new information (including legacy information).

In order to address these three issues, a reuse environment must organize component repositories within a domain ontology while preserving its original characteristics of distribution and heterogeneity, all in a flexible way. This organization can be accomplished through a software layer that provides the integration of different domain repositories (distributed and/or heterogeneous). This software layer can be seen as a particular case of Heterogeneous and Distributed Data Base Systems (HDDS) [4]. One solution in HDDS is the use of the mediation technique [2].
In this paper, we present the adaptation of an integration data layer, based on mediators, to support reuse environments, particularly the Odyssey environment [5]. The main objective of this layer is to integrate information from several domains stored in distributed data sources, in a way that users have transparent and uniform access to these domains while preserving their autonomy. This layer also stores a description of components repositories providing metadata services such as queries on the available components.

2. Mapping a Mediation Architecture into a Reuse Environment

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). 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 on the mediation structure.

![Architecture of the Odyssey Mediation Layer](image)

Figure 1- Architecture of the Odyssey Mediation Layer

The description of each domain (Mediator) is stored on a metadata repository. The metadata represents the domain ontology as well as the description of the components provided by the mediator. Each mediator provides a uniform view of the available components through their representation within a global model. Also, each mediator is responsible for the mapping between the description of the component within the domain and the component itself as stored in the component repository. To handle different formats and representations, the architecture provides a translator module that maps the local description into the global component description model.

Figure 1 presents the Odyssey Mediation Layer that comprises four levels: Interface, Mediation Layer, ORB bus and Translators, which were derived from the mediation model of the HIMPAR architecture [3]. The Interface level is implemented by the Service Manager (SM) which creates and modifies the mediation layer. The Mediation Layer provides the management of each mediator through the Metadata Manager (MM) and also provides the access to mediators though the Query Manager. At the ORB level,
communication between the mediation layer and translators is established through the
CORBA standard services. Finally, the Translator level provides one translator for each
component repository so that it can participate in the Mediation Layer integration model.

Figure 2 presents an example of a mediation layer for a specific application domain,
i.e. The Cardiology Domain. Several mediators are presented as sub-domains, such as
Cardiology and Hospital Administration domains. There are several mediators
responsible for managing specific domains; for example, the Cardiology domain
mediator is specialized (P2) in a mediator that provides detailed information about a
cardiology sub-domain, the domain of Heart attack. The Hospital Administration
Mediator is aggregated (P1) to the Cardiology mediator, 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
data sources that contain reuse components related to its domain.

**Figure 2 - An example of a mediation layer for the cardiology domain**

The KBS data source has the class *Therapy*, whose extension is called *Therapies*, and
the File System data source has the class *Clinical-Treatment*, whose extension is called
*Treatment*, as shown in the ODL definition below. Both data sources store components that
have the ability to, given some patient symptoms, provide a clinical therapy for treating the
patient symptoms, and thus were mapped to the Cardiology Domain Mediator.

```
<table>
<thead>
<tr>
<th>Class Therapy</th>
<th>Class Clinical_Treatment</th>
</tr>
</thead>
<tbody>
<tr>
<td>{extent Therapies}</td>
<td>{extent Treatment}</td>
</tr>
<tr>
<td>{attribute String ComponentDescription;}</td>
<td>{attribute Byte ComponentCode;}</td>
</tr>
<tr>
<td>{attribute Byte Bytecode;}</td>
<td>{attribute String Description;}</td>
</tr>
<tr>
<td>{attribute String InterfaceDescription;}</td>
<td>{attribute String InputDescription;}</td>
</tr>
<tr>
<td>{attribute String sourcecode;}</td>
<td>{attribute String OutputDescription;}</td>
</tr>
<tr>
<td>{attribute String TypeofRequiredInput;}</td>
<td>{attribute String ImplementationLanguage;}</td>
</tr>
<tr>
<td>{attribute String TypeofRequiredOutput;}</td>
<td></td>
</tr>
</tbody>
</table>
```

For the integration of these information, an association class (the Clinical_Therapy
Class) has to be defined in the MM, and stored in the Ontology Model Repository
corresponding to the data types from each data source:
The service structure for the class creation in the MM is:

MM->NewTypeCreation ("Cardiology Domain Mediator", "ClinicalTherapy", "attribute..." )

where the three parameters of the service correspond to char* MediatorName, char*
ClassName, and char* Structure.

To add the data sources KBS and File System to the mediation layer, objects of SM
class model must be instantiated. The syntax for this service for the KBS source is:

SM->ComponentCreation(char* HimparObjectName; char*StructureOfHimparObject);
SM->ComponentCreation("Container", "name = ContainerKBS; Owner = SUS Hospital;
Structures = {{Therapy, Component = Bytecode}, {Therapy, ComponentDescription =
ComponentDescription}, {Therapy, DescriptionofInterfaceInput =
TypeofRequiredInput}, {Therapy, DescriptionofInterfaceOutput =
TypeofRequiredOutput}},Implementational");
SM-> ComponentCreation("Translator", "name = TranslatorKBS; Translator that deals
with Knowledge Based Systems; KBS;{ContainerKBS}");

where HimparObjectName identifies which SM type is being created, i.e., a Mediator,
a Translator, or a Container. The StructureOfHimparObject provides the object
values. It is important to notice that the field Structures, specific of the Container
class, provides a one-to-one mapping between the source attributes and the mediator
attributes of a specific class. In the case of ContainerKBS, the class is named Therapy.
Corresponding objects must also be instantiated for File System container.

Finally, in order to associate the Mediator Clinical_Therapy class with the Therapy
(KBS) and ClinicalTreatment (File System) container's classes, the following service is
provided:

SM->ClassAssociation( "Cardiology Domain Mediator", "ClinicalTherapy",
"{ContainerKBS, Therapy},{ContainerFileSystem, ClinicalTherapy}" );

where the three parameters correspond to: char* MediatorName, char*
MediatorClassName, and char* ContainerClassName.

The following query retrieves information about all containers associated with
Clinical_Therapy class:

Select X.component from X in ClinicalTherapy
where "Cardiac Problems" in X.ComponentDescription;

Therefore, this query provides access to KBS and File System data sources producing
the desired results, i.e., all binary code from components that deals with clinical
therapies for cardiac problems. Such modeling enables the management of data sources
insertions and deletions. Therefore, a large number of data sources can be integrated to
the system without modifications in the Cardiology Domain Mediator. Through the
mediation structure, Odyssey users can search for components in a transparent and
uniform way. In the above example, the users of reuse tools do not have to know where
the components are stored. Moreover, users do not have to query all components
repositories, using the repository query language format (when a query language exists)
to know where the needed components are stored. They do not have do know either how
to access the KBS and File system repositories.

All the complexity for dealing with these heterogeneous repositories is treated by the
mediation layer. Without the layer, Odyssey reuse tools would have to handle these
repositories, increasing the complexity of the Odyssey Environment. Using mediators,
reuse tools can query specifically the mediation metadata, using one single model. The
mappings between mediators metadata and translators redirect and decompose the query
to KBS and File System repositories respectively. Also, the identification of components of the same domain that are in different repositories, as it is the case of objects from Therapy and Clinical_Treatment classes, was detected at the time of the registration of components in the mediation layer. Afterwards this is all transparent to reuse tools.

3. Conclusions

This work addresses the interoperability problem between component repositories in a reuse environment. An integration layer was developed to help the search and identification of suitable reuse components in the Odyssey environment. This layer is based on mediators and ontology to provide the binding of different components to their domain concepts. To assist the identification of related components and their appropriated domain organization, each mediator represents a domain ontology and provides the mapping to their respective components repository.

Mediators provide a uniform view of the available components organized in a domain taxonomy. Domain ontologies are used to help the search for reusable components through the representation of domain semantic concepts. Therefore, this mediation layer promotes domain integration and mechanisms to translate component requests across ontologies. The innovative aspect of our proposal is the use of domain ontologies, for reusable component retrieval, allowing users to express component requests at a higher level of abstraction when compared to keyword based access or component interface based access.

Using this mediation layer, the structure of the reuse environment becomes more flexible, since the existing domain information/component can be easily added to the environment, without conversions from the original information format (format of the information source) to the reuse environment format. Without the mediation layer, the Odyssey reuse tools would have to access directly the repositories, dealing with specific characteristics of each repository. Moreover, when a new repository has to be used, all reuse tools have to be updated for dealing with this new one.

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, 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 retrieval in a reuse environment is innovative. Currently we have 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.

References
