A Multi-Agent System for Domain Information Discovery and Filtering

Regina M. M. Braga *Marcelo N. Costa* *Cláudia M. L. Werner* *Marta Mattoso*
COPPE/UFRJ - Computer Science Department
Federal University of Rio de Janeiro
Caixa Postal 68511 – CEP 21945-970
Rio de Janeiro – Brazil
{regina, mcosta, werner, marta}@cos.ufrj.br

Abstract

The main objective of Domain Engineering is to provide domain information that helps the specification of domain applications. Several Domain Engineering methods organize domain information using different representations, which can be stored in various formats. This paper presents a navigation agent system that provides heterogeneous/distributed access to domain information. Ideas drawn from the field of autonomous agents, user modeling, hypermedia, and mediation are combined into a multi-agent system responsible for discovery and filtering of domain information. An evolutionary model of the user interests and domain ontology are some of the underlying concepts of the agent system that helps users to identify relevant domain information.

Keywords: Agents, Information Filtering and Discovery, Domain Information Reuse, Domain Engineering.

1. Introduction

The ever-growing volume of information on the web has motivated research on information discovery and filtering techniques. Web search engines are not always effective since they are mostly based on keywords search and thus provide broad results.

This filtering must conform to the user profile, providing: i) changes according to user habits and interests; ii) monitoring evolutions on the user profile; and, iii) discover information that can be useful to the user.

Component Based Development (CBD) has received a lot of attention in software development. Domain Engineering (DE) [11] provides a systematic approach to components reuse along all application development phases.

Components must be developed according to a specific DE process, and they can be diagrams, source code, textual descriptions, and binary code, among others. Thus they are available in many different formats. Within a DE support infrastructure that considers CBD [2], it is necessary to provide access to components that can be reused in all phases of an application development within a given domain.

However, the volume of information to be searched can become very large, since it is often necessary to browse through information from different domains, configuring a complex task.

For each domain supported by a DE infrastructure, a set of reusable components must be available comprising the various phases of development. When an application/domain engineer needs to search for domain information, he must know what is relevant for his specific domain/application. Since the volume of information to be handled can become very large, and it is often necessary to consider using information from different domains, the search for relevant and related information can become a very complex task. This is especially true when engineers are not aware of all the available information and/or its various representations.

As mentioned before, it is important that the domain information is made available in all abstraction levels (i.e. conceptual, architectural and implementation) to allow reuse throughout the development cycle. Therefore, the DE search engine must be prepared to browse different types of components depending on the aimed development phase.

This paper presents a domain information search mechanism used within a DE support infrastructure, the Odyssey reuse infrastructure [2] [23], which is based on recent techniques for information discovery and filtering. Techniques such as mediator [19], user model [1], and adaptive hypermedia [18] are used for the specification of a navigational multi-agent system, named Odyssey Search Engine (OSE). The main contribution of this work is to present a search engine that provides identification and filtering of relevant domain information along the phases of CBD within the Odyssey reuse infrastructure.

The paper is organized as follows. In Section 2, we briefly present the adopted technical solutions for domain information discovery and filtering. Section 3 discusses some related works. Section 4 provides an overview of Odyssey-DE process and its corresponding support infrastructure. Section 5 provides details of the OSE architecture, and Section 6 presents our concluding remarks.

2. Technical Solutions for Domain Information Retrieval

The main issue in domain information retrieval is handling information in various representations and different abstraction levels. Some possible technical solutions to this problem are:

1. To use adaptive[1] hypermedia techniques which, based on a user model[2] (profile), can decide how to present information, considering the following requirements:
   i) The user knowledge level on domain terms should be kept in his profile; and
   ii) There should be a way to identify in which level of abstraction the user is searching for components (i.e., conceptual, architectural, or implementation), so that the engine presents him with information that is in the same level.
2. To adapt web search filtering techniques: there should be a way to adapt the user model based on the observation of his navigation activity. For instance, suppose that the user is navigating through documents where certain domain terms appear very frequently, or that he is repeatedly following certain paths. This kind of information should be kept in his profile indicating some possibly relevant information to be used while performing his searches.
3. To use a mediation layer: the user should not need to be aware of various information representation, such as a specific database query language or knowledge representation/query language (e.g., OntoLingua [21]), in order to have access to its contents. He should be able to access heterogeneous or distributed information within a common framework. The situation is complicated when domain information is available in files with varying formats, which is often the case [9]. If these information resources are geographically distant, the search for heterogeneous and distributed domain information will become even more complex. One possible way to solve this problem is to use the mediation technology[3] as proposed by Wiederhold [19], allowing the user to

[1] Adaptive hypermedia systems provide the capacity to perform adaptive document navigation and/or content presentation [18]. The adaptive navigation guides the user through a hypermedia web, suggesting him the most relevant links, or automatically changing destiny links, according to his goals. Adaptive presentation allows that the contents of hypermedia nodes be adapted to user requirements and expectations.

[2] A user model specifies his characteristics, interests and habits, with the aim of improving human/machine communication.

[3] A mediator is a software layer that links distributed bases with heterogeneous data models to the user, presenting him with information in an adequate format.

access domain information in one single way, without having to worry about its locality or representation format conversions.

Within the Odyssey infrastructure [2] [23], the domain information search mechanism adopts a combination of these solutions to provide effective results, meaning relevant information to the user.

3. Related Works

Some works regarding DE support environments can be found in the technical literature [5], [9]. None of these presents an approach for accessing domain information from several domain resources with varying formats, in one single way to the user, as we do in our work. These often restrict reuse to domain information that has been produced within their own environment. Neither do they provide a search mechanism that helps users to find relevant information.

In the Internet search context, several approaches have been proposed for information filtering, based mainly on agent technology. Projects such as Jasper [7], Amathea [15], Hypercontext, Avanti, and Personal WebWatcher (PWW) [14] use different techniques with varying levels of sophistication for the specification of web filters. Most of these approaches are based on HTML documents. However, as mentioned in [13], the structure of HTML documents is not adequate for information search, since it does not provide enough semantic details. The use of XML standard would minimize this problem. Although we currently adopt an OO representation for domain models, these can represent XML documents also. It is possible to include a wrapper for accessing semi-structured data, so that a mediator can execute queries on this kind of data, i.e., HTML and XML documents. Furthermore, although these approaches present effective techniques for information filtering, such as user modeling for information retrieval based on user preferences, machine learning for document processing, among others, none of them deals specifically with domain information. Our approach combines filtering agents with recovery agents that act based on domain ontologies[4]. This aggregates more semantics to information.

The work presented in [17] describes a search engine for the retrieval of reusable code components, such as JavaBeans and CORBA components. The Agora system uses an introspection mechanism for registering code components, through its interface. As a result, this information may not be available when the repository site is not running, or when the information cannot be located. In both cases, the 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 access flexibility to remote component repositories. There is no need for an index phase, and the mediator is able to capture any update that occurs in a remote repository. Besides, the AGORA system only deals with code components. Our approach deals with domain information in all abstraction levels, including code components. Moreover, new information is always associated to domain terms within a given domain ontology, improving its accessibility and reuse. It is well known that reuse possibilities are increased when components are bound to domain concepts [11].

Another important work to mention is the RIG initiative [16], which describes a reuse library interoperability approach. The idea of 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

[4] 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.

RIG model. In this case, it does not provide the flexibility of mediators. 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 is able to search for information based on domain concepts and filtering of relevant information, as we do in our work.

4. The Odyssey Infrastructure and its corresponding method

As mentioned before, the main goal of the Odyssey infrastructure is to provide and access components (domain information) that can be reused in all phases of an application development within a given domain. Therefore, it has been conceived as a framework (Figure 1) where conceptual models, software architectures and implementation models are specified within previously selected application domains.

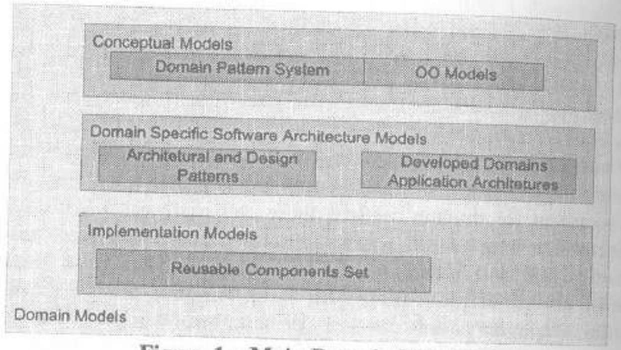

Conceptual Models
Domain Pattern System
OO Models
Domain Specific Software Architecture Models
Architetural and Design Patterns
Developed Domains Application Architetures
Implementation Models
Reusable Components Set
Domain Models

Figure 1 – Main Domain Models

The main conceptual models used by the Odyssey infrastructure are: i) *Domain Use Cases and related OO models*. Use cases are mainly used to capture basic domain functionality in a way that it is possible to derive other OO models. The use case model is the most used model to elicit domain information. In fact, the information is elicited using scenarios[5] that are used to instanciate use case templates[6]. Based on these domain use cases, it is possible to derive other OO models that are essential for a complete understanding of the domain; ii) *Features Model*. It presents, in an abstract level, relationships among domain main concepts and functions. This model is based on the original FODA's features diagram [6], having added to it feature types and the correspondence to ontology as used within information systems research [10].

The *Features Model* provides two main views: the domain concepts model (representing basic domain concepts and relationships among them), and the functional model (presenting, in an abstract level, relationships among domain functionality, which are detailed by domain use cases). However, for a complete understanding of domain concepts, including their synonyms and restrictions, among other information, a structured template similar to HotDraw framework documentation patterns [12] (i.e., a *Domain Pattern*) is used to describe them into detail.

Architectural and Design Patterns [4] [8] together with *Developed Domain Applications Architectures* are used to represent *Domain Specific Software Architecture* (DSSA) models. The *Implementation Model* is represented by the set of corresponding reusable components.

[5] Basically, a textual description of a domain functionality.
[6] Such as the one proposed in [20].

These domain models are specified and further modified according to the activities of the Odyssey-DE process [2]. All diagrams are presented using a modified UML [25] notation. According to Jacobson et al. [11], DE is responsible for the search, analysis, management and formalization of relevant domain information (domain models). Its main objective is to provide a reuse program that promotes the development of domain applications with reduced costs. The DE process is composed of four stages: domain viability analysis, domain analysis, domain design, and domain implementation.

The main objective of the domain viability analysis stage is to verify the viability of the domain to provide a group of reusable components. This viability analysis adopts selection criteria for the domain, attributing weights for each criterion. These criteria should be defined in a way that contemplates organization needs. The domain analysis stage consists in the definition of main domain concepts and functionalities, standing out similarities and differences among them in a high level of abstraction. Models are partitioned based on the main domain functionalities. Concepts form the domain ontology. In the domain design stage, these domain concepts and functionalities are refined, adding architectural considerations to them. Hence, domain specific architectures are established. Finally, in the implementation stage, components specified in the previous stage are coded, having their interfaces for communication with each other defined. In this stage, it is possible to aggregate legacy components to other domain components through the definition of a wrapper.

Domain models can be either stored using a mediation structure, or directly in a local DBMS. This depends on the diversity of domains that need to be tracked, the necessity of integration among them, and/or domain repository heterogeneity. If domain components are stored by the mediation structure, they are preserved in a distributed and heterogeneous way. In this case, the mediation layer is responsible for providing a uniform data representation and manipulation mechanism. The main objective of the mediation layer is to provide the integration of information from various domains that is stored either locally or in distributed data sources, in a way that it is transparent to the user. It is important to notice that with the mediation structure, it is possible to reuse domain information provided by third parties, as considered in our search scenario. In this case, a mediator is responsible for the translation and relationship of the third party domain information with other domain information tracked by the DE support infrastructure. Details about this mechanism can be found in [3] and [22]. However, if there is only one domain, and domain information is stored locally, a DBMS can be used instead. There is no need for a mediation layer in this case.

5. The architecture of Odyssey-Search Engine

Odyssey-Search Engine (OSE) is responsible for domain information (i.e., domain items) search within the Odyssey infrastructure. It is composed of an interface agent (IA), filtering agents (FAs), and retrieval agents (RAs) that search for relevant domain information. Figure 2 presents a diagrammatic view of OSE architecture.

The architecture is divided into three layers:

- An adaptive hypermedia user interface (AI), where search results are displayed according to the user preferences captured by the user model;
- FAs that are able to adapt the user model based on his navigation behavior, and use this to filter the most relevant domain information to be displayed by the adaptive hypermedia;
- RAs search over distributed/heterogeneous domain bases, based on search information provided by FAs. The requested information is returned to the corresponding FA, which again filters out the most relevant one, and passes it over to the adaptive hypermedia to be properly presented to the user.

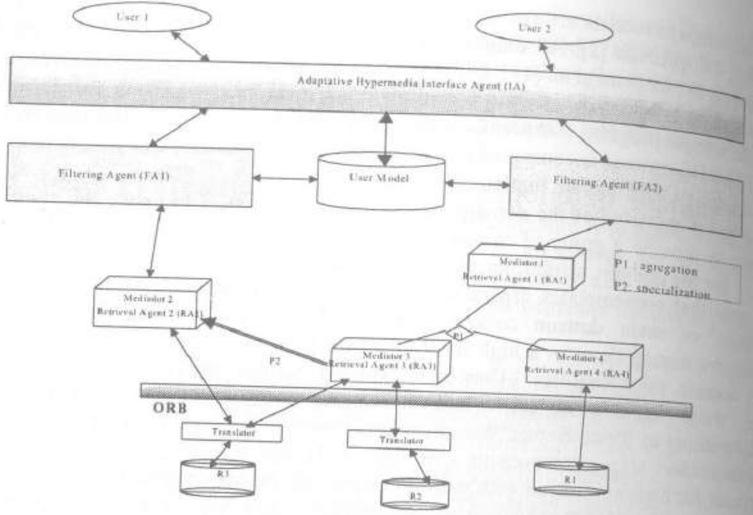


Figure 2 – Basic OSE Architecture

5.1 User Model

In the OSE context, user modeling involves the creation of: *domain categories*, identifying sub-domains; *stereotypes*, classifying users with similar interests, including their preference for certain domain categories; *user models/profiles*, where information such as name, stereotype, and past search information, including their corresponding interest weight[7], are kept. The user model class diagram is presented in Figure 3.

User models can be specified in several ways, such as [7]: direct interviews with the user, adoption of user stereotypes, use of *machine learning* techniques to identify certain behavioral patterns, or use of example based profiles.

OSE combines these techniques to obtain information about the user. First, the user registers as an OSE user by answering a questionnaire (Figure 4) that indicates his knowledge level on domain terms, corresponding stereotype, and interest for certain domain categories. This questionnaire is the only explicit feedback that he needs to provide the system. After that, agents can act without user intervention.

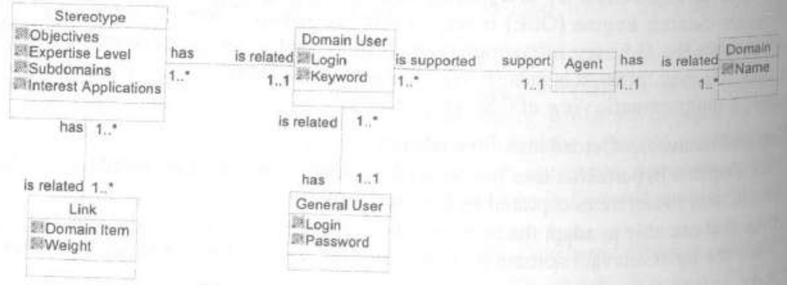
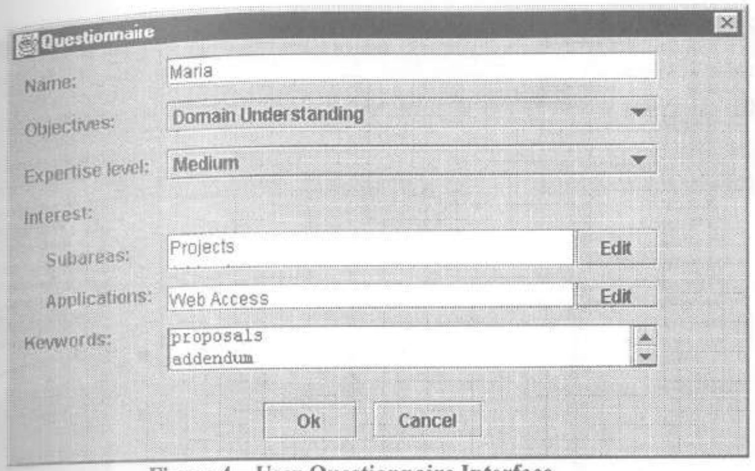


Figure 3 – User Model Class Diagram

[7] This weight is related to the user interest level for a given component. If the user has already examined a given component, a corresponding weight for his interest on it is provided.



Name:	Maria
Objectives:	Domain Understanding
Expertise level:	Medium
Interest:	
Subareas:	Projects
Applications:	Web Access
Keywords:	proposals addendum

Figure 4 – User Questionnaire Interface

OSE questionnaire description is as follows:

a) ***Expertise level***: it indicates the user domain knowledge level. There are three possibilities:

- Low: In this case, the user knowledge about the domain is minimal, i.e., OSE considers that the user never studied the domain. Thus, it is necessary that details about each domain term/component are provided to the user;
- Medium: OSE considers that the user has a basic knowledge about the domain. He probably studied the domain before or developed a simple application within it. In this case, OSE asks at the beginning of his navigation if he wants a more detailed description about the domain;
- High: OSE considers the user a specialist in the domain and does not provide detailed descriptions about domain items.

b) ***Sub-areas***: indicate sub-domains that are interesting to the user.

c) ***Applications***: present applications that were previously developed in the domain.

d) ***Keywords***: OSE will use the words typed by the user as a basis to reinforce algorithm action.

Once the user answers the questionnaire, the agent tries to associate this new user to an existing user profile. This association is based on stereotypes. Each stereotype stores a profile related to a group of users. A user is always related to a stereotype. OSE considers that if a group of users has the same profile, their domain of interest, and consequently their navigation paths, can be the same [16, 19, 23]. Therefore, the questionnaire acts as an indicator of the user stereotype in a way that OSE is able to help in his navigation. Based on the stereotype and keyword validation, OSE can infer navigation paths that should be adequate to the user (see section 5.3 for details).

After registering as an OSE user, he can start navigating, for example, from a context diagram such as the one shown in Figure 5. In this diagram, only the sub-domains (i.e., sub-areas) registered in his profile are displayed. First, a sub-area has to be selected in order to search for related information, based on the corresponding domain ontology (see section 5.4). As the user navigates through the web of information, his profile may change.

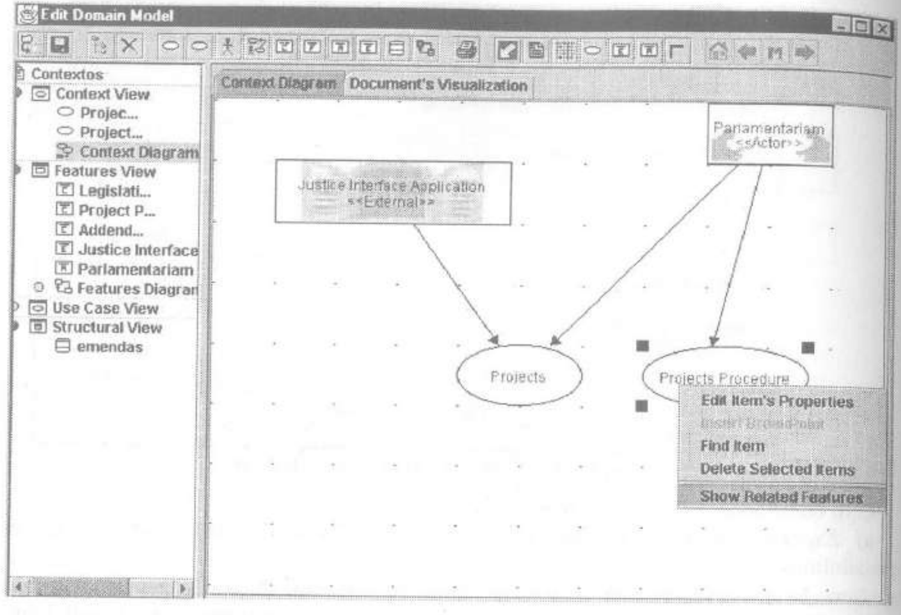


Figure 5 – Initial User Navigation Example

5.2 Interface Agent

In the Odyssey infrastructure, all domain item presentations are based on an adaptive hypermedia interface. Adaptive hypermedia systems perform adaptive documents navigation and/or content presentation [18]. The adaptive navigation guides the user through a hypermedia web, suggesting him the most relevant links, or automatically changing destiny links, according to his goals. Adaptive presentation can adapt the contents of hypermedia nodes to the user requirements and expectations.

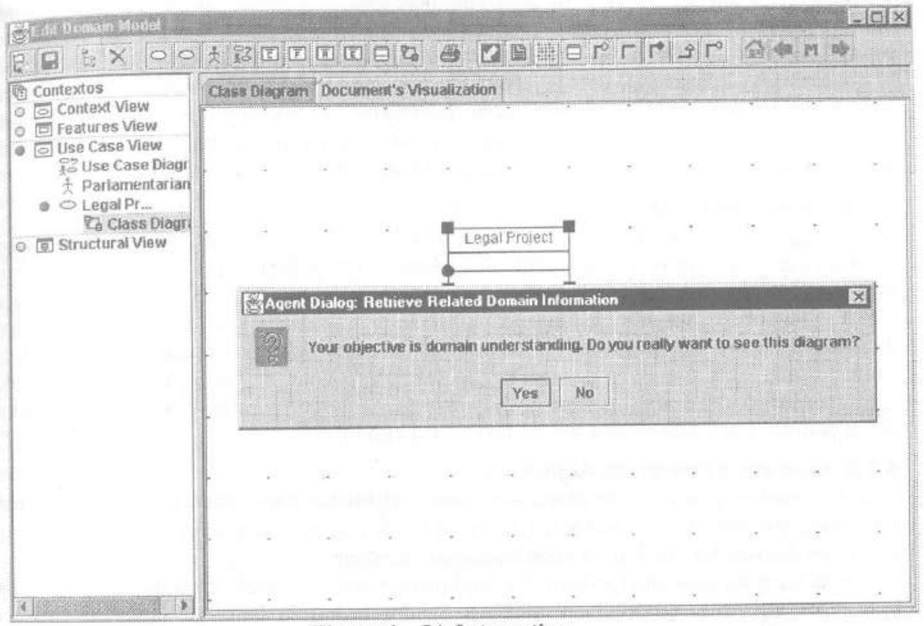
In OSE, the adaptation is achieved by examining the user profile regarding the following items:

- User learning level about domain items (components). This is related to the user expertise level. It is important to notice that this expertise level may change while he is navigating. This aspect determines whether the interface has to be modified in order to show more details for each presented component;
- The component abstraction level that the user is interested in (conceptual, architectural or implementation). This determines if a given component will be presented or not to the user during his navigation.

With successive user navigation, his profile may change. Some examples of this kind of change are given bellow:

- When the user navigates through domain items that represent a given concept and the user does not show any interest about it, the Interface Agent (IA) captures this behavior and alerts him about the importance of this concept;
- When the user is not an expert, for each displayed term, IA adapts the interface and presents a description of the component (Figure 8).

- When the user is only interested in domain understanding and not in application development, details about the architectural and implementation components might not be presented to him (Figure 6).



Your objective is domain understanding. Do you really want to see this diagram?

Figure 6 – IA Interaction

5.2.1 Knowledge Base

A simple representation used in the artificial intelligence field is *if-then* rules. This representation is simple and easy to understand. It can be considered as an individual part or an information unit in a knowledge base. Also, with the if-then rules format, it is easy to add new rules in the base or to change existing ones. IA uses an if-then rules base together with an inference algorithm to process interface adaptation.

The IA rules system is composed of three main elements:

- Knowledge base: the base of facts related to the adaptation process;
- Work memory: the information captured by IA during navigation;
- Inference machine: to process the knowledge base together with the work memory. Specifically, IA uses a *forward chaining* algorithm to generate new facts based on the working memory.

Consider the example: **Rule 1** (*if expertise = low and objective = domain_understanding then developer = "Novice")*, and **Rule 2** (*if developer = "Novice" and Diagram = Feature and Domain_Item = Feature then Action = "Present Details"*). Where rule 1 provides a fact, i.e., "the developer is a Novice". This assertion is used to indicate the action to be taken, i.e., to "Present Details". IA uses this information in order to adapt the interface.

Currently, the adaptations that can be treated by OSE are the ones presented in this section, although the rules system is generic enough to allow new types of adaptations.

5.3 Filtering Agent

The user profile is adapted based on the observation and learning of his behavior while navigating through domain information. *Machine learning* techniques provide the means for "learning from experience", i.e., by observing user searches, navigation paths, and keywords that are often used, it is possible to identify relevant information to the user. Also, it is possible to infer their corresponding interest weight, by considering, for instance, the number of occurrences of these keywords in related domain items, as done by traditional text retrieval techniques, or patterns of navigation paths repeatedly followed by him and/or other users from the same stereotype. This information is used whenever filtering for most relevant information is needed, or deciding which kind of search is to be done.

5.3.1 Prediction Algorithm

While predicting the best navigation path to follow, the filtering agent matches the user keywords and the textual description of each domain item in the navigation path. The domain item that has the greatest number of occurrences of user keywords is considered the best match in this phase, but the occurrences of other domain items are also considered. After that, the filtering agent examines the log of navigation paths (history navigation) followed by similar users. This log is associated to the user stereotype. A log is a link base. Links constitute the model items that were already chosen for stereotype users during the navigation in the domain. Each link owns a weight that indicates a hit rate.

5.3.2 Relevance Feedback Algorithm

In our context, relevance feedback from users establishes the relevance of a domain item for domain process analysis and helps the agent to identify most important items that belongs to a given domain for other users from the same stereotype.

Agent learning uses an algorithm that captures the user feedback, updating its knowledge base. This algorithm is based on relevance feedback and its main advantage is that the analysis of relevance is made in a transparent way. When the agent indicates a model item to the user, and if the user navigates through this item, its link weight is increased by a unit at the corresponding stereotype link base. Next time that the prediction of links is accomplished, this link will have a higher weight in the calculus of the most important links for the user.

Before the execution of the relevance feedback algorithm, the following steps are performed:

1. The interface agent accomplishes the prediction of links (section 5.3.1 above).
2. In the prediction process, the group of items which represents the agent suggestion is indicated.
3. Starting from this suggestion, the user will probably begin a new navigation, and it will be based on the previously agent indicated items.
4. According to this navigation, the relevance feedback algorithm is invoked.

At this stage, it is not necessary that the user interfere in the learning process during domain navigation, that is to say, he does not need to explicitly indicate the model items that are really interesting to him. The agent improves its accuracy through a continuous learning process, without the need of an initial training, or retraining during the infrastructure use by several users. Another important observation is that the algorithm automatically updates the stereotype knowledge base to which the user profile belongs.

5.4 Retrieval Agent

In order to provide an architecture that is able to handle the requirements of component search and retrieval, a Retrieval Agent (RA) was specified based on mediation and ontology

technologies. RA is derived from an HDDS mediator, adding more precision and semantics to it by using ontologies tailored to software component retrieval.

Figure 7 presents an example of a RA configuration for a specific application domain, i.e., the Legislative Domain. Several mediators are presented as sub-domains, such as State Legislative and Municipal Legislative domains. The State Legislative Mediator is aggregated (P1) to the Municipal Legislative, 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. The Federal Justice Domain Mediator may be accessed in cases where the user wants components related to the Justice domain. In Figure 7 we also present a typical RA configuration, with 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. 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.

The Service Manager (SM) stores metadata about mediators, translators and data sources availability, deals with ontological commitments between related mediators (domains). In order to use a given mediator within RA, the administrator has to register the mediator, its related data sources, and translators used by these data sources into SM.

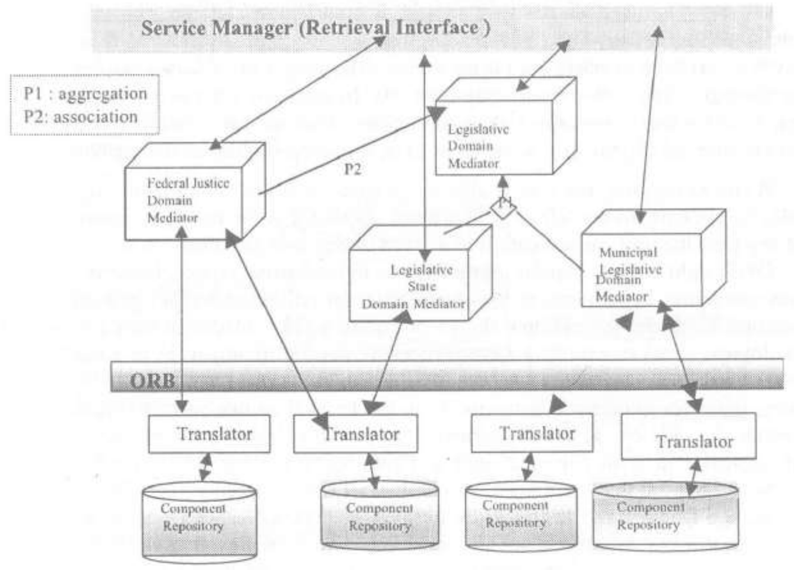

Service Manager (Retrieval Interface)
P1 : aggregation
P2: association
Legislative Domain Mediator
Federal Justice Domain Mediator
P2
Legislative State Domain Mediator
Municipal Legislative Domain Mediator
ORB
Translator
Translator
Translator
Translator
Component Repository
Component Repository
Component Repository
Component Repository

Figure 7 - An example of a typical RA configuration

Each mediator has its own metadata. This metadata represents the ontological model of the domain. This metadata also comprises relationships among ontological terms and components stored in data sources. In order to relate each ontological term with its counterparts in data sources, the mediator manager retrieves related information of data sources from SM, using the ORB bus. The mediator uses the retrieved information to locate and retrieve software components from data sources. Thus, we can associate each ontological term with related components, with the help of a specific translator.

Another important characteristic of RA is the use of domain ontologies 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 of each mediator and associate terms among them. For capturing each ontological model, SM uses the ORB bus, through an IDL interface, to access the specific mediator, retrieving its ontological terms. Therefore, the ontological model, i.e., the Features 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 term being considered. Thus, it is possible to associate ontological terms from multiple domains, providing accessibility for domain information. In a query formulation, SM accesses and retrieves all related mediators, searching for components that fulfill the query semantics.

6. A Navigation Example

Our approach is motivated by a project based on CBD that is being conducted within the Legislative domain. In this scenario, consider the context diagram displayed in Figure 5. This diagram displays sub-areas selected by the user when he/she answers the questionnaire (Figure 4). The user can navigate through related features, clicking on the *Features* option from a pop-up menu. The corresponding features diagram is displayed (Figure 8). The user can now navigate to related use cases and/or class diagrams, if he/she wishes, by clicking on the corresponding option from a pop-up menu. In any case, a RA is activated with the goal of searching for this information, which as mentioned before, can either be in a local base or in a third party base (Figure 10). In the latter case, it is necessary to use the mediation layer.

While navigating, the user is able to go backwards and forwards through his navigation path, by clicking on the left or right arrows, displayed at the top-right corner of the window. At any time the user can see details of a given component by looking at its properties.

OSE might suggest that the user navigates through other related domains. In this case, RA uses synonyms, hypernyms, or hyponyms links for retrieving domain information from other domains. Consider the example shown in Figure 8. The selected feature (in our context, an ontological term) represents a Legal Procedure concept (component in a highest abstraction level). The agent can advise the user, by looking at his profile, past searches and navigation paths, to retrieve domain information from the Federal Justice domain (Figure 7). If the user accepts the advice, RA will retrieve the possible links between the Legal Procedure ontological term in the current domain and ontological terms in the Justice domain.

Now, suppose that there exists an ontological link between a Legal Procedure term in the Legislative Domain and Justice Rules term in the Federal Justice Domain, and that these two ontological terms are related by an hyponym ontological link (Figure 9). In this case, there would be an information loss because the term, Legal Procedure, is more generic than Justice Rule (Justice Rule is an hyponym of Legal Procedure). OSE alerts the user about this information loss and retrieves domain information related to the Justice Rules term to the user. Use Cases, Class Diagrams and Code Components related to Justice Rules term are all made available to the user. The user can, then, use this information in his application.

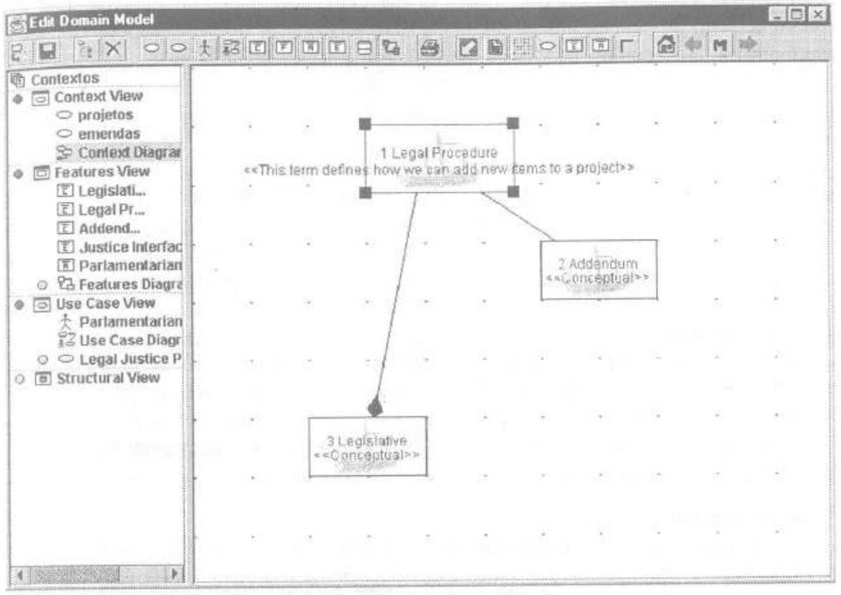


Figure 8– Legislative Domain Navigation Example

In our example (Figure 9), the data source 2 has a binary software component called "New Subject", and the 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 Rules that is mapped to a component named Rule Database.

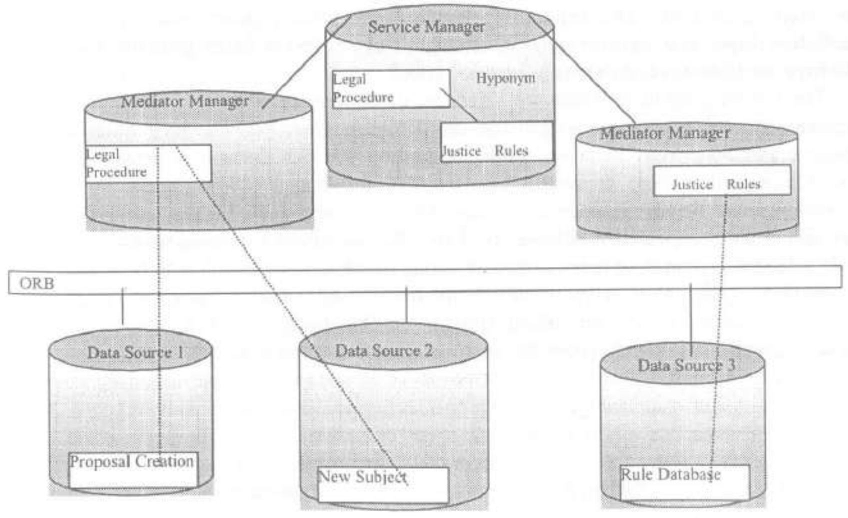


Figure 9 – Mappings between Ontological Terms

These components are made available to the Legislative Municipal Domain through the ontological term in the mediator called "Legal Procedure" that is mapped to the above components in data sources, i.e., data sources 1 and 2.

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

When the 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 the IA accesses the RA interface in order to retrieve components related to the creation of new proposals, it will access components from the Legislative Mediator and the Justice Mediator. It can also select the type of component to be retrieved (components belong to analysis, architectural, codification, or to all phases of development).

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

7. Conclusions

This paper presented a domain information multi-agent search mechanism used within a DE support infrastructure, named Odyssey, which is based on advances in discovery and filtering of web information. Techniques such as user models (profiles), mediators, and adaptive hypermedia were used for composing a navigation multi-agent system.

The main contribution of this paper is to present OSE and its underlying technologies, enhancing in current domain information discovery and retrieval. Although, the search for domain information is not a new subject, its adaptation, using user modeling, intelligent agents, and domain ontologies is innovative.

OSE contribution aspects are: the filtering of domain information, done by the filtering agent layer, where user preferences, past searches, navigation paths, and most used keywords are used to improve and refine the search; and domain information retrieval, using a mediation layer and domain ontologies, to retrieve relevant heterogeneous and distributed information from several domains.

The filtering agent provides the selection of domain information, based mainly on user preferences and experience within the domain. *Machine learning* techniques are used to observe and learn user behavior while navigating through domain information. Also, it is possible to infer their corresponding interest weights, by considering the number of occurrences of words in documents that are frequently seen by the user, or patterns of navigation paths repeatedly followed by him. This information is used whenever filtering for most relevant information is needed or deciding which kind of search is to be done.

Without OSE, the user has to search for domain information using the current available techniques, such as keyword based spiders. In this kind of search, the user is probably presented with a lot of irrelevant information. Besides, even if he finds some interesting site, the available domain information might not be in an adequate format, requiring some kind of conversion. Moreover, there is no helping mechanism that guides him to more interesting related information, as we do in our work.

The Odyssey reuse infrastructure is in constant improvement and so is the OSE tool. By analyzing the current OSE architecture, we can list some enhancements that can be done, such as:

- Our filtering strategies can suffer from the effects of positive feedback loops. Our approach presents the highest value component first (see Figure 8, Legal Procedure component). This can cause a positive feedback loop. The reason is that many users have a natural tendency to select the first entry (in our case, the Legal procedure Component). This phenomenon would tend to skew the prediction metadata over time, because of the physical position, not leading to a real value judgment.
- The textual analysis algorithm can also be improved. Currently, it can do only matches based on identical words. It could also do matches based on word stems

We are currently working on these and also in providing an infrastructure that allows users to navigate through the Internet. Another future improvement is to estimate the cost for accessing distributed domain information using RA, by estimating the cost of distributed query retrieval.

An operational prototype, implemented in Java and C++, which codifies hypermedia connections, mediators, translators, and CORBA communication protocol is now available OSE uses the services of an OODBMS, named GOA++ [24], for storage of local data.

Acknowledgements

The authors would like to thank CNPq and CAPES, for their financial investment in this work.

References

[1] Eftihia Benaki; Vangelis A. Karkaletsis;, Constantine D. Spyropoulos; "User Modeling in WWW: the UMIE Prototype"; In: Proceedings of the workshop "Adaptive Systems and User Modeling on the World Wide Web workshop"; at the Sixth International Conference on User Modeling; Chia Laguna, Sardinia, June 1997

[2] 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, March, 1999, pp.50-57.

[3] 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 (TOOLS-30 USA'99) Conference, Workshop on Component-Based Software Engineering Process, IEEE CS Press, Santa Barbara, California, August 1999, pp.542-546.

[4] Buschmann F. et al.; Pattern-Oriented Software Architecture: A system of patterns; John Wiley, 1996.

[5] Chan, S; Lammers, T.; "Reusing a Distributed Object Domain Framework"; In: Proceedings of the Fifth International Conference on Software Reuse; Canada, 1998.

[6] Cohen, S.; Feature-Oriented Domain Analysis: Domain Modeling; Tutorial Notes; 3rd International. Conference on Software Reuse; Rio de Janeiro, November 1994.

[7] DAVIES, N.J.; WEEKS, R.: Jasper: Communicating Information Agents for WWW at http://www.labs.bt.com/projects/knowledge/jaspaper.htm

[8] Gamma E. et al; Design Patterns: Reuse of Object Oriented Design; Addison Wesley, 1994.

[9] Gomaa, H et al; "A Knowledge-Based Software Engineering Environment for Reusable Software Requirements and Architectures"; Automated Software Engineering, 3 (3/4): 285-307, August 1996.

[10] Guarino, Nicola; "Formal Ontology and Information Systems"; In: N. Guarino (ed.) Formal Ontology in Information Systems;. IO Press, Italy 1998.

[11] Jacobson, I.; Griss, M.; Jonsson, P. ; "Software Reuse: Architecture, Process and Organization for Business Success";, Addison Wesley Longman, May 1997
[12] Johnson, R.; "Documenting Frameworks"; In: Proceedings of OOPSLA'93, 1993
[13] Sean Luke et al.; "Ontology-based Web Agents"; In: Proceedings of First International conference on Autonomous Agents, 1997.
[14] Mladenic, Dunja; Machine Learning on non-homogeneous, distributed text data, PhD Thesis, University of Ljubljana, 1998.
[15] Moukas, Alexandros; Amathea: Information Filtering and discovery Using a Multiagent Evolving System, Master Thesis, MIT, 1997.
[16] RIG; "Reusable Library Interoperability Group" at http://www.asset.com/rig/, 1996.
[17] Seacord, R.; Hissan, S.; Wallnau, K, "Agora: A Search Engine for Software Components", Technical Report CMU/SEI-98-TR-011, August 1998.
[18] Staff, Christofer; "HyperContext: A Model for Adaptive Hypertext"; In: Proceedings of the Sixth International Conference in User Modeling, UM'97; Vienna, New York: Springer Wien New York, 1997
[19] Wiederhold, Gio; Jannink, Jan: "Composing Diverse Ontologies"; prepared for IFIP Working Group on Database 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)
[20] Cockburn, Alistar, Use Case Templates, at http://members.aol.com/acockburn
[21] Wielinga, B., Van de Velde, W., Schreiber, G. and Akkermans, H. (1993) Towards a Unification of Knowledge Modeling Approaches; David, J-M., Krivine, J-P, and Simmons, R. (Eds.) Second Generation Expert Systems. Springer Verlag, Berlin.
[22] Braga, R.; Mattoso, M.; Werner, C; "Using Ontologies for Domain Information Retrieval", In: Proceedings of DEXA 2000, DomE Workshop, 4-8, September 2000.
[23] Werner, C.; Braga, R.; Mattoso, M.; Murta, L.; Costa, M.; Pinheiro, R.; Oliveira, A.; "Infra-estrutura Odyssey: estágio atual", XIV Simpósio Brasileiro de Engenharia de Software, Caderno de Ferramentas, João Pessoa, October 2000 (in portuguese).
[24] Mattoso, M.; Werner, C.; Braga, R.; Pinheiro, R.; Murta, L.; Almeida, V.; Costa, M.; Bezerra, E. Soares, J.; Ruberg, N.; "Persistência de Componentes num Ambiente de Reuso", XIV Simpósio Brasileiro de Engenharia de Software, Caderno de Ferramentas, João Pessoa, October 2000 (in portuguese)
[25] Fowler, Martin; Analysis Patterns - Reusable Object Models - Addison Wesley Publications, 1997.