Chapter 7

Application: The Web Information Mediator

This chapter describes the Web Information Mediator (WIM), a specific MAS application that uses the ORCAS framework and is implemented over the the ORCAS e-Institution. WIM is a configurable meta-search application to find bibliographical references in the Internet. WIM is based in a collection of domain-independent cooperative information agents, while the current application domain is medicine. This chapter describes this application, and some extra work performed concerning the interoperability of the ORCAS e-Institution.

7.1 Introduction

The process of information search and aggregation on the Internet is essentially a process of *mediation* between the goal of the user and the information resources that are distributed and heterogeneous. Therefore, an application dealing with such information processing tasks can be seen as a Web Information Mediator (WIM). The overall goal of WIM is to provide a mediation service for information search tasks of a professional user in some domain. A mediator is an agent that offers an added value to the information accessed in other sources [Wiederhold, 1992]. Typical services offered by a mediator include selection of information sources, information retrieval, ranking and fusion of information from different sources.

WIM has been developed and specified according to the ORCAS framework, and it is deployed over an ORCAS e-Institution (implemented over the NOOS Agent Platform). WIM is a configurable application according to Definition 4.15, and as such it has the following properties:

- WIM functionality is hold by a library of tasks and capabilities provided

by agents and specified in a domain independent manner. The application domain is characterized by a collection of domain models characterizing the domain knowledge. However, WIM adds like wrappers to the ORCAS e-Institution architecture, which are used to agentify external information sources.

- Problems are specified as a collection of problem requirements and problem data. A specification of problem requirements includes the name of the task to be solved, domain models characterizing specific domain knowledge, preconditions stated to hold, and postconditions to be satisfied after solving the problem.
- There is a clear separation between the problem solving agents, which are the providers of the capabilities, and the domain knowledge, which is defined independently. The domain models characterizing the application domain are specified for any particular problem as part of the problem requirements.
- WIM applications are configured on-the-fly for a particular request by decomposing the problem task into subtasks, binding capabilities to tasks, and verifying that the assumptions of the capabilities are verified by the domain models specified in the problem requirements.
- WIM on-the-fly applications are designed, operationalized and executed according to ORCAS model of the CPS process (S5.2), which is made up of four subprocesses: Problem Specification, a Knowledge-Configuration, Team Formation and Teamwork.

In few words, WIM is a configurable application that exploits the ORCAS framework to customize agent teams on-demand, according to stated problem requirements. The separation of tasks and capabilities from the domain is an architectural pattern that aims at maximizing the reuse of agent capabilities. The matching relations allow to verify whether a set of domain models characterizing the application domain satisfies the knowledge assumptions of a capability. As a result, WIM agents can be reused over new application domains by specifying the new domain knowledge as a collection of domain-models, and building appropriate ontology matchings when there is an ontological mismatch between the concepts used to specify the new domain and the concepts defined in the WIM ontology.

The WIM mediation process can be seen as a sequence of adaptation steps: to begin with, WIM adapts the user request (a consultation) to generate domain queries, using knowledge from the specific application domain, like medicine; secondly, WIM agents have to adapt the queries expressed in terms of the application domain to the features of specific information sources; lastly, the information retrieved from several queries and different sources is adapted again to obtain a global result to the user request.

An important distinction in WIM is established between a *domain query* and a *source-query*. On the one hand, a domain-query is a description of the type of

information to search from an abstract, conceptual viewpoint, and is expressed in terms of application domain. On the other hand, a source-query is used to specify a query to a specific information source, in terms of the search strategies and the filters allowed by that particular source.

A second consideration is the paradigmatic distinction between the concept of "relevance" in classical information retrieval (IR), and the richer conceptualizations currently in use for intelligent information agents [Carbonell, 2000]. The canonical concept of relevance in IR is a test where the results of a query by a retrieval engine are compared to a gold standard provided by a human expert that assesses false positives and false negatives. The problem of that approach is that *relevance* is independent from the purpose of the specific user in posing a query. Therefore, current research also tries to establish a *utility measure* that is task-dependent: retrieved items are evaluated to determine in which degree satisfy the intended purpose expressed in the user's query [Carbonell, 2000].

WIM takes into account both considerations: on the one hand, WIM allows the user to specify an information search problem in terms of a domain query, and then it uses domain knowledge to transform the domain query into source specific queries; on the other hand, the result of applying each query can be interpreted according to a notion of *utility* to the task and not only of relevance with respect to the content of the query.

Modern information systems should manage or have access to large amounts of information and computing services. The different system components can conduct computation concurrently, communicating and cooperating to achieve a common goal. These systems have been called *Cooperative Information Systems* [International Foundation on Cooperative Information Systems, 1994]. A major goal of this field is to develop and build information systems from reusable software components. This goal can be achieved by assembling information services on-demand from a collection of networked legacy applications and information sources. WIM is a specialization of that general class of systems in the field of MAS, for it is built by connecting the capabilities provided by a society of agents to some domain knowledge specified as a collection of domain-models. WIM is designed as a configurable MAS according to the ORCAS framework, and has been implemented as a society of information agents participating in an ORCAS e-Institution (developed upon the *Noos Agent Platform*). The WIM application is configured by selecting and configuring the capabilities brought by the WIM information agents, and linking them to a medical domain knowledge (medicine in general, and Evidence Based Medicine in particular). The configuration of an information task (a task-configuration) is operationalized by forming a team of agents with the capabilities encompassed by the configuration, and imposing team members a commitment to that configuration. WIM is configured on-the-fly for each request to solve a problem, and the configuration is built according to the kind of problem and the preferences of the user.

To sum up, WIM tasks and capabilities are reusable because they are defined in a domain independent manner, thus the same agents can be used to build a new application with a new domain knowledge.

This chapter begins introducing the WIM approach to information search (§7.2) and briefly describing the WIM architecture (§7.3). Following, the main elements of the WIM application are described at the knowledge level: the information search ontology (§7.4) first, the specification of tasks and capabilities (§7.5) second, and the domain-models (§7.6) third. Besides, the chapter devotes an entire section to usage aspects such as the interfaces to the application (§**??**), another section to exemplify the use of the WIM library (§7.7), and another one describing an interlibrary application (§7.10) made in cooperation with other partners in the IBROW project.

7.2 The WIM approach to information search

The vast amount of information available in the Web causes some serious problems to the users when looking for information. It is difficult to find all the relevant information, for it is distributed among several information sources. In addition, the information retrieved is rarely ranked according to the users utility criteria. Meta-search (Figure 7.1) is one of the most promising approaches to solve the first problem. If the single search-engines store only a portion of all the existing information about some particular domain, several search engines should be queried to increase the search coverage. But often in practice, meta-search is under-exploited; existing meta-search engines do not combine results from the different single engines, neither they rank the information retrieved, or the ranking mechanisms are quite poor. Our approach overcomes these two limitations. The WIM application tasks allow to rank documents retrieved from information sources that originally do not perform any ranking; and it allows also to aggregate information retrieved from different search engines. The originality of our vision is that we exploit the filtering capabilities of existing search-engines, thus ranking is achieved with a little computation and storage cost.

These are, in brief, the main requirements we pose to the information search process:

- it should be possible to rank information retrieved from sources that are not able to rank information by themselves;
- to retrieve and combine information coming from multiple, heterogeneous, sources;
- to aggregate the rankings of information retrieved for several queries, so as to bring a unique, overall ranking; and
- to rank information according to the users utility, enabling for an easy modification or customization of the utility criteria.

WIM approach to information search can be described as an *adaptation* process with three main stages: query adaptation, information retrieval and aggregation. *Query adaptation* refers to the process of elaborating the user consultation to better fulfill his interest, as well as adapting queries expressed in terms of

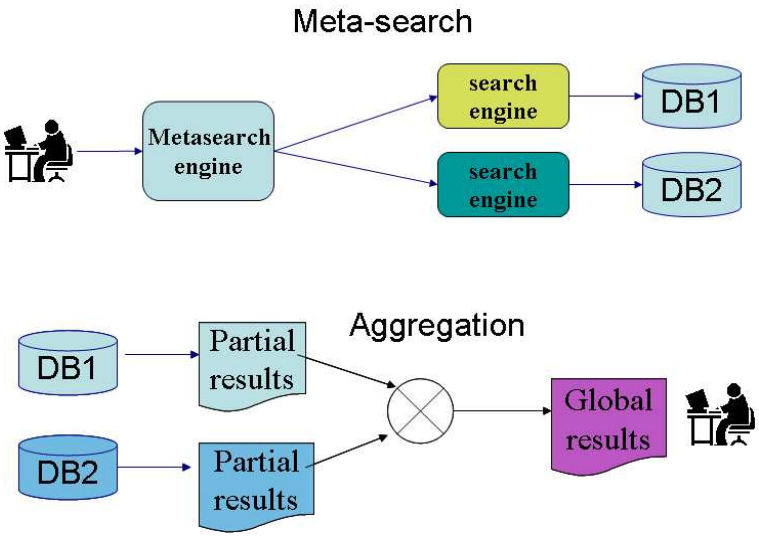

Figure 7.1: Meta-search and aggregation

a domain language (independent of any specific information source) to the language used by particular information sources. Once the retrieval is performed, the results from different queries are aggregated for each source, and finally the results for each source are aggregated again to obtain a unique result for the user consultation.

WIM approach to information search is based on *query weighting* and *numerical aggregation operators* [Gómez and Abasolo, 2003]. Query weighting refers to the process of assigning weights to queries generated according to some domain knowledge, while numerical aggregation operators are used to merge information retrieved from different information sources or different queries. Aggregation allows to combine the different ranking obtained by bits or items of information obtained for different queries, so as to obtain a unique ranking or score for each retrieved item. This approach allows to rank items retrieved from sources that originally do not give any ranking, and to define user-oriented utility measures simply by defining the appropriate knowledge categories (§7.6).

7.2.1 Adaptation of queries

WIM subscribes to the well known approach of representing queries as keyword vectors. This decision is justified because nowadays professional databases could often be accessed through the use of a web-based search-engine, whose queries are made of *keywords* belonging to a particular domain. In addition to keywords, WIM queries allow specifying search *filters* as optional constrains for narrowing the retrieval process. An ontology on bibliographic data has been used to model

the kind of filters allowed by professional bibliography search-engines, like *publication type*, *language* and so on.

There are two types of query: domain-query and source-query (§7.4.2). Domain queries are expressed in terms of the application domain, like medicine, while source queries are customized for a particular information source. In our implementation, the application domain is medicine, and the type of information retrieved is about bibliographical references. Therefore, WIM performs two types of adaptation:

- *Query elaboration* is performed at the domain level, using application domain knowledge to adapt queries to a particular user interest, within a particular domain, like medicine bibliography. Domain queries are elaborated using domain knowledge, like synonyms, hypernyms and hyponyms provided by a thesaurus, which can be used to obtain semantically equivalent queries, and to either generalize or specialize a query.
- *Query customization* is performed at the source level, using descriptions of information sources (Web-based search engines) to customize a query for a particular information source. Source queries are generated by translating keywords and filters from the domain level ontology into the *search modes* and filters supported by a particular information source.

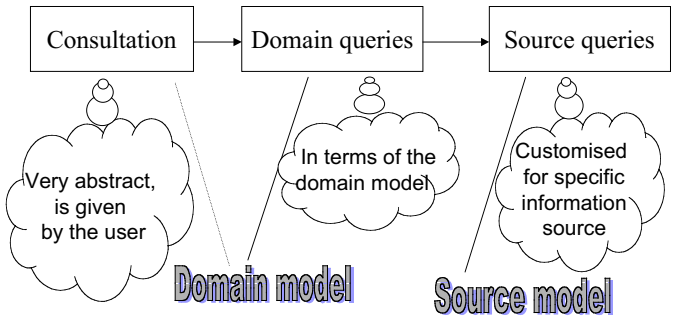

Figure 7.2: Domain and source query elaboration

Figure 7.2 sums up the idea of the query adaptation process at the two levels: domain level and source level.

1. The consultation introduced by the user is expressed as a domain-query, afterwards it is expanded into a collection of new domain-queries, using domain knowledge and utility criteria to elaborate queries. Resulting queries are weighted according to the similarity of the new domain queries to the user consultation.
2. Domain queries are transformed into source queries customized for specific information sources, using domain knowledge about the information

sources.

In our approach to query adaptation, new queries are not only enriched with domain knowledge, but they are also weighted according to the strength of the relationship between the original query and the new query (which depends on the type of query adaptation as well as on specific information represented as domain knowledge). For instance, query adaptation at the domain level (query elaboration) uses semantic similarities to generate new queries, while query adaptation at the source level uses information on the search modes supported by an information source and the type of filters applied. Once the query adaptation finishes, the selected information sources are queried through the use of wrappers, which are responsible for adapting the queries represented in terms of the WIM ontology to fit the particular interface of each information source. Queries generated during the query adaptation phase allow to score the items retrieved even though the information source does not give any ranking. Ranking can be straightforwardly applied because the queries are weighted in advance, during the adaptation process, thus their answers can be considered as inheriting the weight of the query as an assessment of relevance or utility.

A more formal description of the query weighting approach used by WIM information agents is provided as an Appendix in B.

7.2.2 Aggregation of results

The answers to all the queries are the items retrieved from different information sources, ranked according to the weights of the queries they are a result to. These items are combined to obtain a unique set of items, where repeated items are eliminated, keeping only one instance of each item with a unique, overall ranking.

Ranking synthesis is achieved by applying some aggregation operator. Aggregation is a kind of merging where the rankings assigned to different occurrences of an item are combined using an aggregation operator to obtain a unique ranking that summarizes the utility or relevance of each item for the user. A numerical aggregation operator is necessary because during the query adaptation process queries are weighted with numerical values. Hence, the results of a query (the retrieved items) inherit the weight of the query. If the queries are weighted according to some *utility* criteria rather than relevance, then the results are ranked taking into account these utility criteria. Notice that the same item may be retrieved as a result of several queries, and each query may have a different weight. Therefore, the aggregation process has to compute an overall score for each retrieved item based on the evidence degree contributed by the weights of the queries in which each item is retrieved. For this purpose, four numerical aggregation operators have been implemented as capabilities in the the library: the *arithmetic-mean*, the *weighted-mean*, the *Ordered Weighting Average (OWA)* and the *Weighted OWA* [Torra, 1996].

Sometimes, the term "fusion" is used to refer to the aggregation of items retrieved from different information sources. In our framework, the same pro-

cedure is used to aggregate items retrieved from a unique search-engine can be used in fusion. In this case, each source is assigned a weight expressing the reliability of that source or other kind of "goodness". A query customized for multiple search-engines is weighted using a combination of the weight assigned to the query during the query elaboration stage, and the weight assigned to the source in the source description.

7.3 WIM architecture

The WIM configurable application has been specified according to the ORCAS framework, both at the knowledge and the operational levels (i.e. using the ORCAS ACDL), and is configured through the institutional agents provided by the ORCAS e-Institution (Chapter 6).

WIM is based on a collection of domain-independent information agents registered in the ORCAS e-Institution as Problem Solving Agents (PSA), plus a repository of domain knowledge accessible by these agents.

We can distinguish three groups of agents in the WIM application:

- *Institutional agents* implementing the internal roles of the ORCAS institutional framework: Librarian, Knowledge-Broker and Team-Broker. These agents are common for any ORCAS-based application, they just offer the infrastructure to build a configurable application. We have built one agent for each role: there is one Librarian, one Knowledge-Broker and one Team-Broker.

- *Problem Solving Agents* equipped with capabilities to perform information search and aggregation tasks. These agents can participate in the institution by adopting the PSA role and registering their capabilities to a Librarian agent. From now on we will refer to the set of tasks and capabilities registered in WIM as the Information Search and Aggregation (ISA) library.

- *Personal Assistants* specialized in the WIM information tasks. These agents interact with the user (or a software agent) to obtain a specification of the problem, and bring back the results to the user (or agent).

In addition, there are several pseudo-agents providing connectivity to external agents and to non agent resources as well, namely: wrappers, FIPA-mediator and WWW-mediator.

- *Wrappers* are used to agentify external resources, like Web-based information sources (Pubmed, IGM-Healthstar, IGM-Medline, and iSOCO wrappers) and external domain knowledge (MeSH wrapper).

- The *FIPA-mediator* provides connectivity to external agents willing to access the WIM application.

- The WWW-mediator provides connectivity to an http protocol, and thus it supports Web based access to the WIM application.

However, the WIM library needs some domain models satisfying its knowledge requirements to become a complete application. We have specified as domain models some knowledge bases containing the knowledge on medicine and bibliographic data required by WIM capabilities to be applied. This knowledge is stored within a shared repository that can be accessed by any agent playing the PSA role.

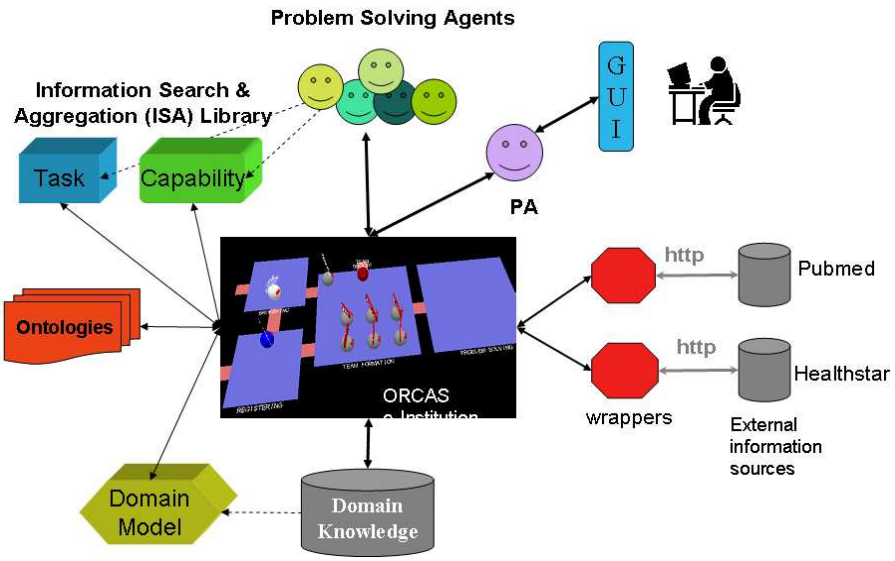

Figure 7.3: WIM architecture and interoperation

Figure 7.3 sums up the main elements of the WIM architecture integrated through the ORCAS e-Institution. There is a library on Information Search and Aggregation that is composed by the capabilities and tasks registered by several information agents (Problem Solving Agents), represented using some ontologies; there is a repository of domain knowledge characterized by domain-model; and there are some wrappers allowing agents to interoperate with external information sources. Moreover, there is a Personal Assistant agent responsible for processing mediating between the user and the application. The ORCAS e-Institution is the central element, constituting a middleware layer where agents meet to interact according to the shared social knowledge (ontologies, communication language and interaction protocols) provided by the institution. The ORCAS e-Institution provides an added value to both providers and requesters through a collection of middle agents that mediate between requesters and providers: Librarian, Knowledge-Broker, Team Broker, and Personal Assistant.

WIM interface to external agents is based on the idea of services. Each service defines a type of operation that can be requested by other agents, and is specified by an interaction protocol and a data format specified according to the FIPA proposals. WIM services specialize the ORCAS services to handle concepts from the the Information Search and Aggregation (ISA) ontology, as described in Appendix F.

7.4 The Information Search and Aggregation Ontology

Ontologies are extensively used through WIM including the following:

- the Knowledge-Modelling Ontology (KMO) at the knowledge level;
- the ontologies defined by the ORCAS e-Institution, which are the Teamwork Ontology and the Brokering Ontology; and
- the ontology used to specify the components in the ISA-Library (the ISA-Ontology) and the ontologies used to specify the domain-models used in WIM.

The concepts of the Knowledge-Modelling Ontology are described in Chapter 4 and Appendix A. The ontologies used within the ORCAS e-Institution are described through several sections of Chapter 6. This section describes the ontologies used to specify the ISA-Library, that is to say, the vocabulary used to specify the features characterizing the tasks and capabilities in the WIM application. Recall that this vocabulary is represented using the Object Language, and we are using of Feature Terms (§4.3).

The ontologies and the components expressed in terms of those ontologies are both represented using the NOOS [Arcos, 1997] representation language. There are several reasons endorsing this decision. First, the ORCAS KMF provides and architectural framework that is neutral about the content language (the Object Language). Second, there is an implementation of an agent platform in NOOS that is compatible with FIPA, so describing the ISA-Library in NOOS facilitates the development of an agent based system for the WIM application. Morevoer, the NOOS agent platform has some facilities to translate XML-based information and is an object-centered representation language; therefore NOOS is compatible with RDF-based representations like those used in describing Semantic Web Services (e.g. DAML-OIL) and by the Protégé editor suite used in Knowledge Modelling and Ontology Engineering.

The approach taken here has been to depart as few as possible from the core assumptions of the ORCAS KMF while using NOOS in a easy and understandable way. For this purpose, we have designed a few macros on top of the NOOS language; these macros are used to generate the NOOS code in the agent-based implementation of the ISA-Library. The point here is to make agents use ORCAS concepts as part of their content language in the agent communication language.

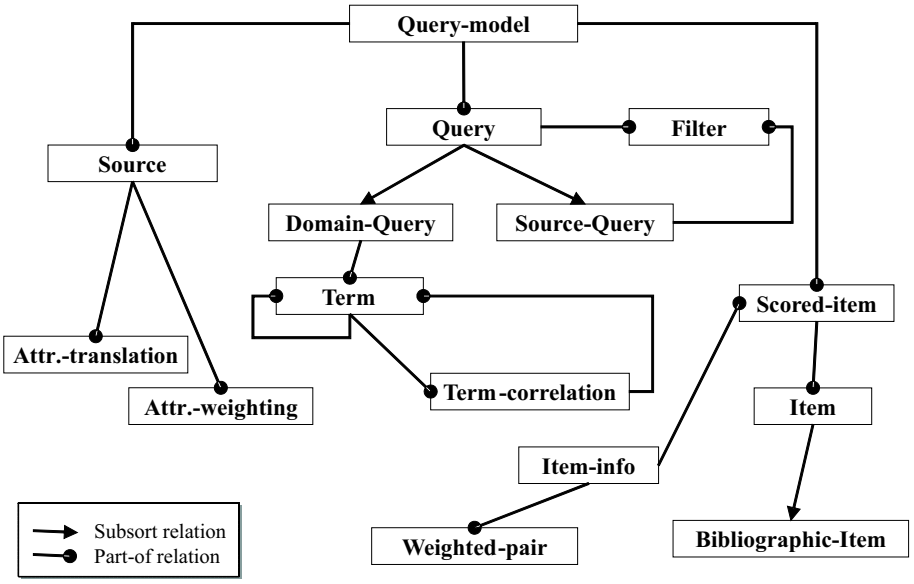

Figure 7.4: Overview of the ISA Ontology

The ISA-Ontology consists of two different parts: one encompasses the concepts used as signature elements and is used to describe the inputs, outputs and knowledge-roles, and the other encompasses the formulae and is used to specify the preconditions, postconditions and assumptions.

There are two sorts that are not subsorts of other sorts: the sort FT-Signature-Element, which is used to specify signature elements in the Object Language of Feature Terms, and the sort FT-Formula, which is used to specify the formulae (realize that these sorts are refinements of the abstract sorts defined by the Knowledge Modelling Ontology, named Signature-Element and Formula).

The main sorts of the the ISA-Ontology used to specify signature elements in the WIM library, and the relations among these sorts are drawn in Figure 7.4. These sorts are described step-by-step in the following subsections, while the formal specification is included in Appendix D. The concepts used to specify formulae are not described as a separate entity; instead, some of them are introduced as needed when describing the tasks and capabilities in the ISA-Library (§7.5).

7.4.1 Items

An *item* (Figure §7.5) is the informational unit used during the information search tasks. It is defined in very generic terms, so as to be refined for different purposes, as follows:

The sort item has the following features:

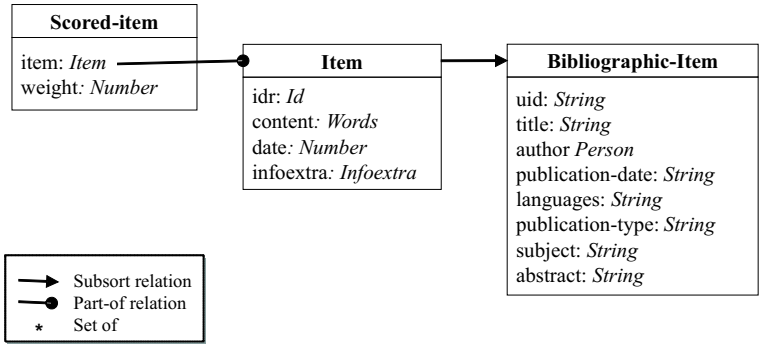

Figure 7.5: Sort definitions of Item, Scored-item and Bibliographic-item

- *Identifier (Id):* it is used as a reference to the piece of information embodied by the item. An item exists initially with a value in its *Id* slot and the remaining slots empty. As long as the search processes take place, the rest of fields are gradually fulfilled.

- *Content:* it corresponds to the actual piece of textual information (e.g. lists of words) that will be gathered from external sources. Its format depends on the source it comes from. For example, a typical content in Internet is an HTML page (and following this Internet analogy, the *Id* would correspond to the URL) whereas in a Data Base, a content will correspond to a specific record and the ID would be the index number.

- *Date:* it is necessary to keep track of the moment that an item of information is obtained or updated. Their values are numerical (they represent number of seconds) so that numerical operators can be easily applied to compare two dates.

- *InfoExtra:* There is additional information used to check if a content should be updated yet or not. For example, this field includes a code for any problem obtaining the content and the language of the content.

The sort Item can be refined to deal with specific types of information; specifically, we are interested in searching bibliographic information, so we have defined a Bibliographic-Item sort that introduces bibliographic data within the specification of an item. Figure §7.5 shows the sort definition for the sorts Item and Bibliographic-item, which is defined as a subsort of the former. The sort Bibliographic-item adds some slots to those inherited by the Item sort, namely a unique identifier (UID), a title, a set of authors, a publication date, a set of languages, a publication type, a subject (usually specified as a set of *keywords*), and an abstract.

A *scored-item* is a pair that associates an item with a number which represents its *score* with respect to the user consultation. The score of an item may

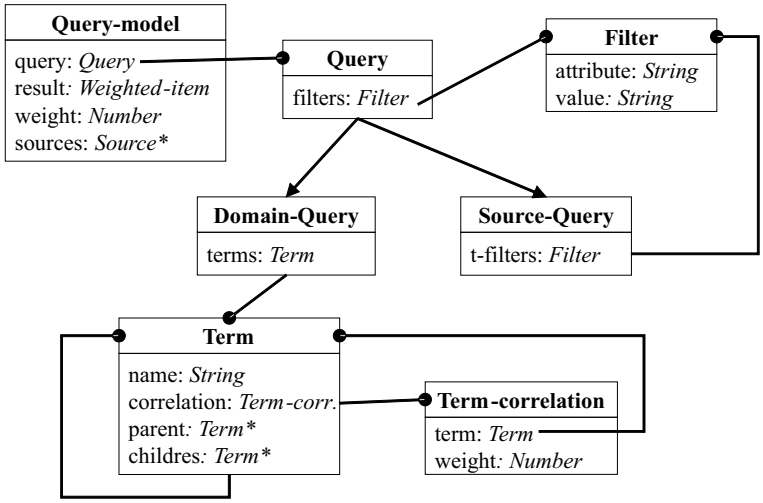

Figure 7.6: Sort definitions of Query, Filter, Term, and Query-model

be given as the result of a query to a source or as the result of an aggregation process over the results of several queries.

7.4.2 Queries, Filters and Terms

The information about what is being searched on behalf of a user is expressed in this ontology by queries, filters, terms and categories. Figure 7.6 shows these sorts and their relationships.

Queries

The general sort query represents the user's consult. There are several types of queries that share only a feature, called *filters*.

The user's consults are specified as instances of the sort Domain-query that in addition to filters contains a list of terms and one category.

When the domain query is translated to the concrete vocabulary of the source to be accessed, i.e. it is *customized*, the query is of sort source-query. The information in a domain-query is translated to *filters* and *t-filters* in a source-query.

Filters

Filters are constraints over objects of the search. They are represented by the sort filter that has two features:

- an *attribute*, which specifies the name of a filter to constrain the search of information; and

- a *value* for that attribute, to be satisfied by the the results to retrieved.

For instance, let us suppose that our goal is to search in the medical database Medline for documents published after 1990. A filter of a domain query will contain as *attribute* "begin year" and as *value* "1990".

Terms

Terms are words pertaining to an ontology used in a query. There are relations between terms (in a ontology) that embody domain knowledge and, as such, they are located in a domain model that uses this ontology. These relations between terms are included in the features *parent* and *children* of sort Term (Figure 7.6).

The feature *parent* represents the relation of a term with other more general terms. The *children* feature represents the relation of a term with more specific terms. The feature *term-correlation* represents the relation between two terms. The sort Term-correlation has two features : *term* and *weight*. *Weight* is the correlation degree of *term* with the current term. When the weight is 1 it means that both terms are synonyms.

Categories

A category is an intensional definition of a class of search objects in the context of a particular domain ontology. For instance, we have specified some categories concerning clinical medicine such as *diagnosis*, *therapy*, and *clinical guidelines* (Figure 7.12).

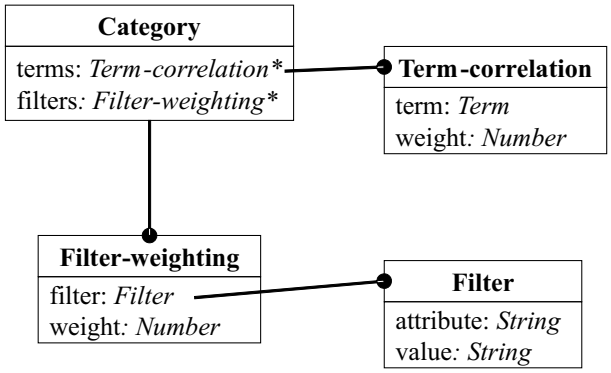

Figure 7.7: Sort definition of Category

A *category* (Figure 7.7) express its intensional meaning as a collection of correlated *terms*, which are specified as elements of sort Term-correlation, and a collection of filters, which are specified as elements of sort Filter-Weighting. A Filter-Weighting is a pair composed of a *filter* and a *weight* .

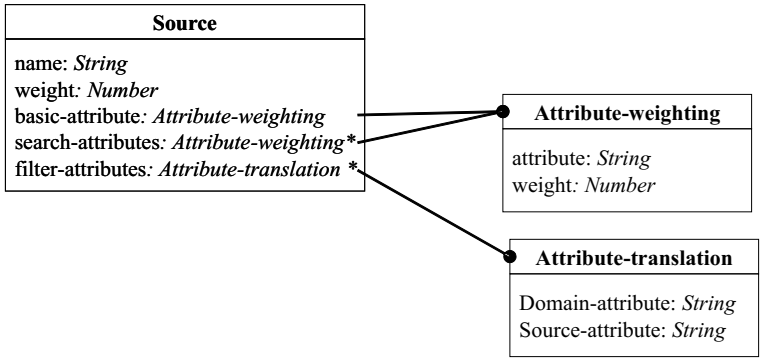

Figure 7.8: Sort definitions of Source, Attribute-weighting, and Attribute-translation

7.4.3 Sources

We consider a source as composed of two parts: the content of a database (for instance Medline or HealthStar) and a particular search engine that can be used to access that content (for instance PubMed or IGM). A source is modelled using the sort source that has the following features:

- *Name*: that represents the name of a source.
- *Weight*: a weight assessing the reliability of a source.
- *Search-attributes*: the set of attributes allowed within a query the source. These attributes may be record fields of a database or search modes of the search-engine. Search modes indicate accesses that are combination of several fields. For instance, the attribute "subject" of the IGM method searches a word both as a MesH term and in the abstract.
- *Basic-attribute*: represents the basic or default search mode of a source For instance, the basic attribute of the PubMed-Medline is called "all", which means that the search is performed using all the available search fields, thus it is the more general way of searching Medline.
- *Filter-attributes*: is a set of attributes that can be applied in order to limit the search. Some examples of filter attributes are the kind of publication, date, language, etc.
- *Content*: The name of the database. For instance, Medline that is a database on medical bibliographical data.

The features *name*, *search-attributes*, *basic-attribute*, and *filter-attributes*, are referred to the method that allows to access the database.

The values in *search-attributes* are of sort Attribute-weighting, which has two features: *attribute* and *weight*. The *weight* indicates the semantic importance of

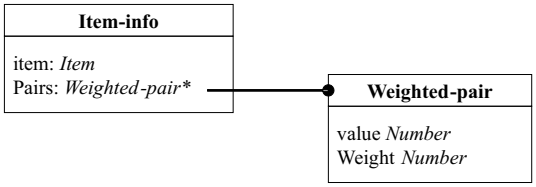

Figure 7.9: Sort definition of Item-info

the *attribute*. For instance, in the Medline information source the weight of the feature *title* is greater than the weight of the feature *abstract*. This means that the same keyword is more important when it appears in the title than when it appears in the abstract.

The feature *filter-attributes* is specified by elements of type Attribute-translation (see Fig. 7.8), which is composed of a *domain-attribute* and a *source-attribute* (both of sort String), and specifies a mapping schema between the domain ontology and the source ontology.

7.4.4 Query-models

The *Query-model* sort (Figure 7.6) encapsulates a query and all extra information to be associated to a query during the performance of the task Information Search, which includes the following features:

- *Query:* the query we are talking about (§7.4.2)
- *Source:* the source to which this query is addressed (§7.4.3)
- *Result:* a set of *scored-items* (§7.4.1) returned by the *source* as answer to *query*.
- *Weight:* the weight assessing the utility or relevance of the *query*
- *Children:* the set of query models related with the *query*. For instance, the query model representing the user's consult is related with query models containing elaborated queries.

7.4.5 Item-info

The aggregation capabilities need all the information about an item in order to calculate a global score for that item. This information is represented by the sort Item-info, which is composed of an *item* and a collection of *pairs*. The feature *pairs* (see Figure 7.9) contains pairs *value-weight* where *value* is the importance of the item and *weight* is the importance of the query that has produced the retrieval of the item.

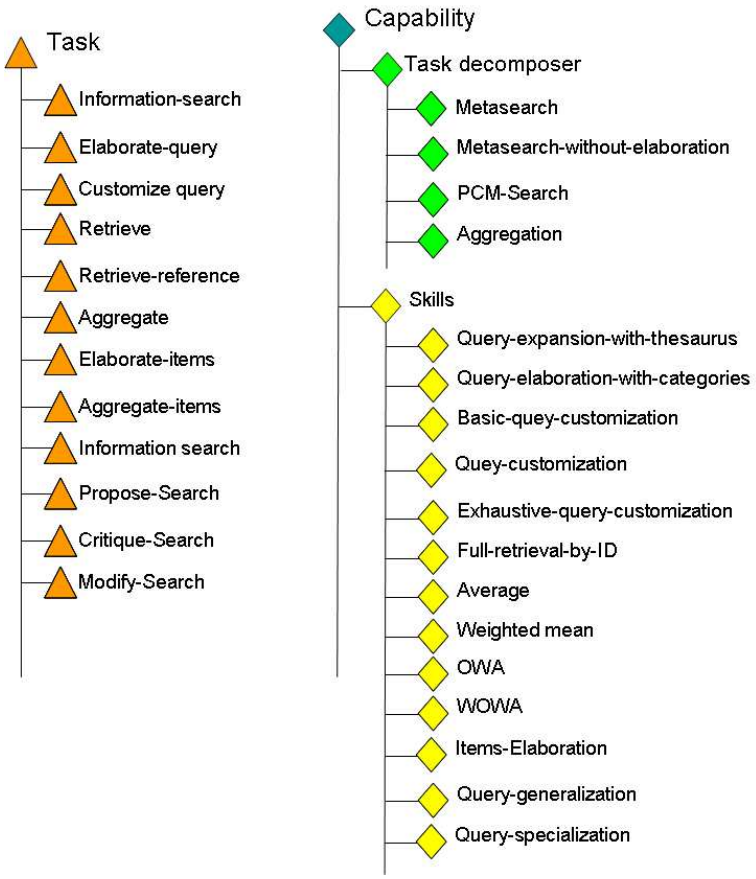

Figure 7.10: Hierarchy of components in the WIM library

7.5 The WIM library

The WIM library is composed of a collection of tasks and capabilities concerning information search and aggregation of information in the Internet. Figure 7.10 shows the hierarchy of components in the WIM library, including 14 tasks, 10 skills, and 5 tasks decomposers. These components are all instances of a sort in the Knowledge-Modelling Ontology, whether it is the Task, Skill, or Task-Decomposer sort.

Tasks are decomposed into subtasks by tasks-decomposers following a hierarchical task/subtask decomposition schema. The top-level task is called Information-search, and have the goal of obtaining a set of items answering a user's consult. Commonly, the user's consult can be expressed as a set of keywords and filters, and several kinds of results are possible (e.g. documents, Web

pages, images, etc.). Within this section the different tasks and capabilities of the ISA-Library are described in terms of the ontology described in the previous section.

Before to go through the specification of components in the library some notes on the Object Language are pertinent. We have considered two formalisms as the Object Language: Lisp-like predicates (*predicate* $?var_1$ ... $?var_N$) and Feature Terms (see §4.3). Although we have used only the Features Terms formalism in the final implementation of WIM due to pragmatic consideration, some examples of the predicate-based representation will be provided at some points for illustration purposes.

7.5.1 Information Search task

```
(define (Task :id Information-Search)
  (ontologies ISA-Ontology)
  (input-roles
   (define (var)
     (name 'consult)
     (sort Domain-Queries))
   (define (var)
     (name 'available-sources)
     (sort Source)))
  (output-roles
   (define (var)
     (name 's-items)
     (sort Scored-Item)))
  (competence
   (define (Competence)
     (postconditions
      SATISFY-CONSULT
      ))))
```

The input for the task Information Search is a consultation by the user, specified as a Domain-Query and a collection of information sources, specified by elements of sort Source. The output is a collection of elements of sort Scored-Item and the goal is to obtain items satisfying the user consultation, which is represented by the formula SATISFY-CONSULT, which is a subsort instance of the sort Formula-FT (from now onwards formulae identifiers are specified using capitalized characters).

There are several capabilities that are suitable for the Information-search task, where suitable means "being able to solve", as established by a *task-capability matching* relation (Definition 4.1). Specifically, there are 4 capabilities suitable for the task Information-Search, and all of them are task-decomposers:

1. Metasearch,
2. Propose-Critique-Modify-Search (PCM-Search),
3. Metasearch-with-source-selection, and
4. Metasearch-without-elaboration.

We are going to describe here the Metasearch and the PCM-Search capabilities. The Metasearch capability is a task-decomposer that decomposes the task Information-Search in four subtasks: Elaborate-query, Customize-query, Retrieve and Aggregate.

```
(define (Task-Decomposer :id Metasearch)
  (name "Metasearch")
  (ontologies ISA-Ontology)
  (input-roles
   (define (var)
     (name 'consult)
     (sort Domain-Queries))
   (define (var)
     (name 'available-sources)
     (sort Source)))
  (output-roles
   (define (var)
     (name 's-items)
     (sort Scored-Items)))
  (competence
   (define (Competence)
     (postconditions
      SATISFY-CONSULT
      )))
  (subtasks
   Elaborate-query
   Customise-query
   Retrieve
   Aggregate))
```

Recall that a task-capability matching is defined at the knowledge level as a combination of signature and specification match (Definition 4.1):

$$match(T, C) = (T_{in} \geq C_{in}) \ \wedge \ (T_{out} \leq C_{out}) \ \wedge \ (T_{pre} \Rightarrow C_{pre}) \wedge (C_{post} \Rightarrow T_{post})$$

The former relation is specialized for the Feature Terms Object Language using subsumption as the inference mechanism (Definition 4.4):

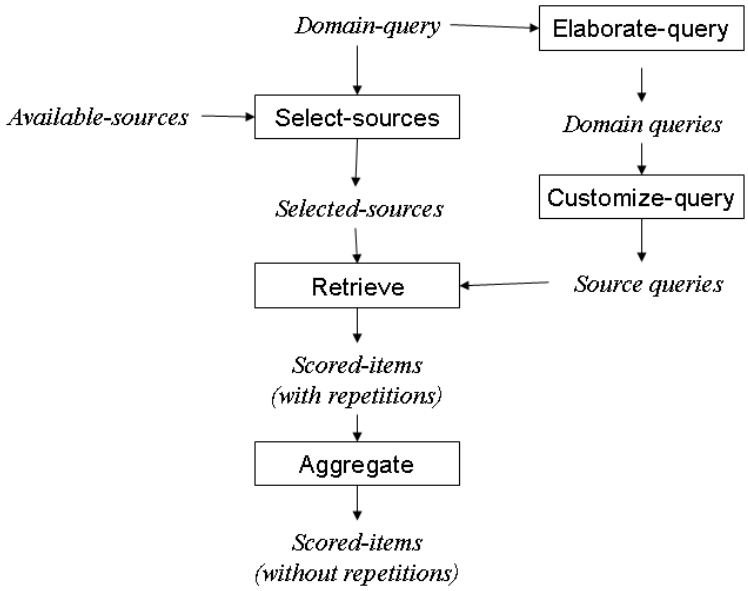

Figure 7.11: Overview of the capability Metasearch-with-source-selection

$$match(T, C) = T_{in} \sqsubseteq C_{in} \wedge C_{out} \sqsubseteq T_{out} \ \wedge C_{pre} \sqsubseteq T_{pre} \wedge T_{post} \sqsubseteq C_{post}$$

The former relation holds between the Information-Search task and the Metasearch capability; specifically, there is an exact match rather than a plug-in match between them, that is to say: the input, output, preconditions and postconditions of the tasks are exactly the same than those of the capability.

The capability Metasearch-without-elaboration is similar to Metasearch, but omits the task Elaborate-query; moreover, the Metasearch-with-source-selection capability is also similar to Metasearch but introduces a new task dealing with the selection of a subset of information sources from the set of available sources.

However, the capability PCM-Search introduces three subtasks: Propose-Search, Critique-Search and Modify-Search. The first and the third of the former tasks are similar to the task Information-Search, thus they can be solved by any of the two Metasearch capabilities.

Figure 7.11 shows the subtasks introduced by the task-decomposer Metasearch-with-source-selection and the data flow among subtasks, which is represented by the sorts of their inputs and outputs.

The task Elaborate-query takes as input the user's consult (a domain-query embedded within a query-model) with the goal of generating new queries so as to increase the recall and the precision of the original query. The idea of this task is to enrich the user's consult using semantic relationships between terms in the query and terms within a knowledge repository such as a thesaurus.

The queries resulting of the Elaborate-query task may be asked to all the available sources or only to a subset of them. For this reason there is a subtask called Select-sources, which takes a collection of sources (available sources) as input and has the goal of determining the subset of those sources where it's worthwhile to search information for a particular consult (selected sources).

The Customize-query task embodies the process of adapting the domain-queries obtained by the Elaborate-query task to the idiosyncratic features of a specific information source. As a result, a set of source-queries are generated as output and further used as input of the *Retrieve* task together with the set of selected-sources resulting of the Select-sources task.

The Retrieve task is performed on query-by-query basis, for it takes a single source-query and a source as inputs. The result of the Retrieve task is a set of scored items retrieved from that source (embodied within the *query-model* containing the *source-query*). Since that task may be performed several times, several collections of items would be retrieved, probably involving repeated apparitions on the same item with a different weight. The next task deals with this multiplicity of gathered information.

The Aggregate task has the goal of integrating all the information about an item retrieved as a result of different queries, perhaps obtained from different sources. The input of the Aggregate task encompasses all the scored-items contained within the query-models resulting of the many performances of the Retrieve task. Each *query-model* has a set of *scored-items* in the *results* slot, i.e. the items retrieved from one source and their associated weight. The goal of the Aggregate task is to eliminate repetitions and to obtain an overall weight for ranking each of the retrieved items —detecting whether two items refer to the same information and using the capabilities on aggregation (see §7.5.8). Thus, the output of this task —that is also the output of the information-search task— is a single set of scored-items.

In the following sections the subtasks of the task-decomposer Metasearch-with-source-selection are described, as well as some related capabilities (some capabilities suitable for the those subtasks).

7.5.2 Elaborate-query task

```
(define (Task :id Elaborate-query)
  (name "Elaborate-query")
  (ontologies ISA-Ontology)
  (input-roles
   (define (var)
     (name '?consult)
     (sort Domain-Query)))
  (output-roles
   (define (var)
     (name '?elab-queries)
     (sort Domain-Query)))
```

```
(competence
 (define (Competence)
   (postconditions
    ELABORATE-CONSULT))))
```

The Elaborate-query task takes the user's consultation (*?consult*) as input, specified as a Domain-query, and has the goal of generating new domain queries (*?elab-queries*) as output, so as to increase the recall and the precision of the original query. The idea of this task is to enrich the user's consult using semantic relationships between terms in the query and terms within a knowledge repository such as a thesaurus. The elaboration of a query is useful when the user wants to search documents about a topic that can be expressed using different, alternative keywords and filters. In order to go through the Elaborate-query task, a capability generates new queries taking into account some domain knowledge (e.g. a *thesaurus*) without the user having to declare it. The output of the Elaborate-query task is a set of elements of sort Domain-query.

An alternative specification of the Elaborate-query task is provided below using a predicate-based representation in place of the Feature Terms formalism.

```
(define (Task :id Elaborate-query
  (ontology ISA-Ontology)
  (input (?consult :sort Query-model))
  (output (?elab-queries :sort Query-model))
  (preconditions (subsumes domain-query ?consult.query))
  (postcondition (forall (?qm in ?elab-queries)
                   (elaborated-from ?qm.query ?consult.query))))
```

There are four capabilities, all of them skills, suitable for the Elaborate-query task:

1. Query-expansion-with-thesaurus,
2. Query-expansion-with-categories,
3. Exhaustive-query-expansion-with-thesaurus,
4. Exhaustive-query-expansion-with-categories,

The capability Query-expansion-with-thesaurus is a skill that generates new queries in which some terms —the keywords— are replaced by synonyms or related terms, and the queries are weighted according to the semantic similarity between the original term and the new term (represented with a correlation coefficient). The capability Query-expansion-with-categories does not replace terms by other terms; this method generates new queries by adding new keywords and filters to the user consultation according to a category (a topic) selected by the

user. The knowledge used to enrich the queries is obtained from a collection of predefined categories or topics. The exhaustive versions of the former capabilities do the same work but they generate a greater number of queries by using more complex variations of the original query (replacing more than one keyword or filter per query).

```
(define (Skill :id Query-expansion-with-thesaurus)
  (ontologies ISA-Ontology)
  (input-roles
  (input-roles
   (define (var)
     (name '?consult)
     (sort Domain-Query)))
  (output-roles
   (define (var)
     (name '?elab-queries)
     (sort Domain-Query)))
  (competence
   (define (Competence)
     (postconditions
      ELABORATE-WITH-THESAURUS)))
  (knowledge-roles
   Thesaurus))
```

The Query-expansion-with-thesaurus skill takes a domain-query as input (*?consult*) and produces a collection of domain-queries as output (*?elab-queries*); the goal is to build new goals by changing or adding elements of the original query, which is represented by the formula ELABORATE-WITH-THESAURUS. This formula means that the new queries are modifications of the input query using knowledge from a thesaurus (synonyms and related words), so a knowledge-role of sort Thesaurus is specified within the *knowledge-roles* slot. If we compare the specification of this skill with the specification of the Elaborate-query task we realize that they match, though it is not an exact match but a plug-in match.

A task-capability matching holds between the Information-search task and the Query-expansion-with-thesaurus skill, though there is not an exact competence match, but a plug-in match. Specifically, the formula ELABORATE-CONSULT specified as a postcondition of the task and the formula ELABORATE-WITH-THESAURUS satisfy the subsumption relation established by a task-capability matching, for the sort of ELABORATE-WITH-THESAURUS is a subsort of the sort of ELABORATE-CONSULT (Figure 4.14), and thus the condition $T_{pos} \sqsubseteq C_{pos}$ holds.

An alternative version of the former skill is provided below using a predicate-based representation.

```
(define (Skill :id Query-expansion-with-thesaurus
  (ontology IS-Ontology)
  (input (?consult :sort Query-model))
  (output (?elab-queries :sort Query-model))
  (knowledge thesaurus)
  (preconditions (subsumes domain-query ?consult.query))
  (postconditions
    (forall (?el-q in ?elab-queries)
      (and (elaborated-from ?el-q.query ?consult.query))
           (uses-synonyms ?el-q.query ?consult.query))))
```

Again, one can verify that a task-capability matching relation holds between the Information-search task and the Query-expansion-with-thesaurus skill, though a renaming of *?el-q* to *?qm* is required. If we compare the postconditions of both specifications we found that the postconditions of the capability imply the postconditions of the task, since they are more specific (notice the capability incorporates an additional clause in the postconditions with the predicate *(uses-synonyms ?el-q.query ?consult.query)*.

7.5.3 Select-sources task

Given a domain query and a set of available sources, the goal of the Select-sources task is to determine a subset of the input sources useful for the query.

There are several capabilities for solving the Select-sources task. The simplest one is Ask-user; this skill shows the available sources to the user and lets him choosing the subset to be used for retrieving information. This capability is used when the user has some knowledge about which sources may be more useful.

Another capability for solving the Select-sources task is the Case-based-source-selection capability, which is based on using similarity measures. The capability Case-based-source-selection imports a domain model that has a case base and a similarity measure. Each case represents a domain query that has been successfully asked to a specific source.

7.5.4 Customize-query task

The Customize-query task deals with the adaptation of domain queries to the features of a specific information source. The goal of the customize-query task is to translate a query expressed using the terms of the domain into a collection of new queries expressed in the terminology of a particular source. The customization focuses on the attributes of the filters used in the input query. There are two possible cases: 1) an attribute is allowed by the source, and 2) an attribute has to be translated to the ontology used by the source. In case 1) no translation is needed, and in case 2) the capability uses knowledge storing a description of the source to know which are the mapping schemes between the domain vocabulary and the source vocabulary. One of the attributes of a

source is the collection of search modes allowed. The point is that the keywords in the domain query may be searched using different modes or strategies, hence more than one source query can be generated for the same domain query using different search strategies and trying alternative filters. The purpose of trying different search strategies and filters is to obtain more information to rank the items retrieved. For instance, if search information for a query using a very general search mode and then repeats the same query using a more restrictive search mode, one can compare the items appearing in both queries to determine which results are more relevant to the query (the ones appearing as a result of the two queries, see §7.7).

```
(define (Task :id Customize-query)
  (ontologies ISA-Ontology)
  (input
   (define (var)
     (name '?query)
     (sort Domain-Query)
   (define (var)
     (name '?source)
     (sort Source)))
  (output-roles
   (define (var)
     (name '?s-queries)
     (sort Source-Query)))
  (competence
   (define (Competence)
     (postconditions
       CUSTOMIZE-DOMAIN-QUERY))))
```

The input of this task consists of a query of sort Domain-query (as those resulting from the Elaborate-Query task) and one specific source from those ones specified in the user's consultation or selected by the Select-source task. The output is a collection of source queries (*?s-queries*) which are customized for the input source, as expressed by the formula CUSTOMIZE-DOMAIN-QUERY.

There are three capabilities, all of them skills, that match the Customize-query task:

1. Query-customization,
2. Exhaustive-query-customization, and
3. Basic-query-customization.

```
(define (Skill :id Query-customization)
  (ontologies ISA-Ontology)
  (input-roles
   (define (var)
     (name '?query)
     (sort Domain-Query)
   (define (var)
     (name '?source)
     (sort Source)))
  (output-roles
   (define (var)
     (name '?s-queries)
     (sort Source-Query)))
  (competence
   (define (Competence)
     (postconditions
      NON-EXHAUSTIVE-CUSTOMISATION
      )))
  (knowledge-roles
   Source-Descriptions))
```

7.5.5 Retrieve task

The goal of the Retrieve task is to effectively retrieve from a source a set of items satisfying the input query. The Retrieve task takes a source-query and a source as input, and produces a set of scored-items as output (specified within a Query-Model, in the *results* slot), such that an item's score expresses the degree the item satisfies the input query.

```
(define (Task :id Retrieve)
  (name "Retrieve")
  (ontologies ISA-Ontology)
  (input-roles
   (define (var)
     (name 'query)
     (sort Source-Query)
   (define (var)
     (name 'source)
     (sort Source)))
  (output-roles
   (define (var)
     (name 'result)
     (sort Query-Model)))
  (competence
```

```
(define (Competence)
  (postconditions
   SATISFY-QUERY))))
```

There is only one capability suitable for the Retrieve task, the Wrapper-based-retrieval skill, with a specification matching exactly the specification of the Retrieve task. We have decided to use Internet as the place to look for information; specifically, we focus on Web-based search engines dealing with medical bibliographic databases, such as *Medline* (accessed through the *Pubmed* search engine) and *Healthstar* (accessed through the *Internet Grateful Med* search engine).

From this point of view, the real retrieval capabilities are the Web-based retrieval engines, and the Wrapper-based-retrieval capability is a mere bridge to a source-specific wrapper that agentifies the Web-based search engine by handling the query and parsing the html results to obtain a structured representation of the information.

7.5.6 Aggregate task

```
(define (Task :id Aggregate)
  (name "Aggregate")
  (ontologies ISA-Ontology)
  (input-roles
   (define (var)
     (name 'q-models)
     (sort Query-Model)))
  (output-roles
   (define (var)
     (name 's-items)
     (sort Scored-Item)))
  (competence
   (define (Competence)
     (postconditions
      AGGREGATE-ALL))))
```

The goal of the Aggregate task is to achieve a coherent and unique set of scored items from all the items retrieved from all the queries originated from the user's consult. The input for this task is a collection of query-models, each one containing the results concerning a single source-query. The output is a set of scored items, where all the information concerning the same piece of information is aggregated to obtain a unique scored item summing up the scores assigned to the different references to the same information unit.

There is a capability suitable for this task, the Aggregation task-decomposer. This task decomposer introduces two subtasks: Elaborate-item-infos and Aggregate-item-infos.

The Elaborate-item-infos task (see §7.5.7) transforms the information contained in a collection of query-models into a collection of item-infos, each one gathering all the information about a single piece of information originally distributed among several query models. The purpose of this task is to represent the information in a format more appropriate for the numerical aggregation operators implemented in the ISA-Library (see §7.5.8). The Aggregate-item-infos subtasks receives (iteratively) the information about an item (an item-info) and uses a mathematical function like a mean to compute a single score for that item. For this reason, the capability Aggregation performs an iteration of the Aggregate-item-infos subtask over the set of item-infos that are the output of the Elaborate-item-infos subtask.

```
(define (Task-Decomposer :id Aggregation)
  (name "Aggregation")
  (ontologies ISA-Ontology)
  (input-roles
   (define (var)
     (name 'q-models)
     (sort Query-model)))
  (output-roles
   (define (var)
     (name 's-items)
     (sort Scored-Item)))
  (competence
   (define (Competence)
     (postconditions
      AGGREGATE-ALL
      )))
  (subtasks
     Elaborate-items
     Aggregate-items))
```

Notice there is an exact match between the Aggregate task and the Aggregation capability.

7.5.7 Elaborate-item-infos task

```
(define (Task :id Elaborate-items)
  (name "Elaborate-items")
  (ontologies ISA-Ontology)
  (input-roles
   (define (var)
     (name 'q-models)
```

```
    (sort Query-Models)))
  (output-roles
   (define (var)
     (name 'item-infos)
     (sort Item-Infos)))
  (competence
   (define (Competence)
     (postconditions
      ELABORATE-ITEM-INFOS))))
```

The Elaborate-item-infos task has the goal of grouping all the information about the same unit of information (e.g. the same bibliographical reference) that is distributed among several query models as a result of retrieval information several times. The objects encompassing all the information on a single unit of information are created as instances of the sort Item-info. The input for this task is a set of query models containing a set of scored items each one, and the output is a set of item-infos.

There is only a capability suitable for this task, the Items-elaboration skill. This capability operates by taking the results (the scored items) of a collection of queries (represented as query models) and looking for those items referring to the same piece of information, like a bibliographic reference. As a result of the query expansion process carried out for the Elaborate-query task and the Customize-query task, the same piece of information may have been retrieved for different queries, thus appearing as different items. The goal of this capability is to detect whether different scored items refer to the same piece of information in order to eliminate redundancy. However, the aggregation of the different scores assigned to the same piece of information is performed by a capability suitable for the Aggregate-item-infos task. The Items-elaboration skill keeps a set of item-infos, one for each new piece of information, and iteratively compares each scored item with the set of item-infos to decide whether they refer to the same piece of information; if the answer is yes, the information on the item is added to the item-info, else a new item-info is created. Item infos encompass two types of information: weights and scores. Weights are obtained from the weights assigned to the query models where an item appears, whilst scores are assigned during the retrieval process by the information source or assigned on a default basis: a maximum score of 1 for each apparition of an item within a query-model, and 0 for each absence. In conclusion, an item info is a collection of pairs composed of a value and a weight (v_i, w_i), the former representing the relevance assigned to an item by the retrieval engine, and the later expressing the combined utility and relevance estimated during the elaboration and the customization of a query.

7.5.8 Aggregate-item-infos task

The Aggregate-item-infos task is a purely mathematical task and the capabilities that can solve this task are capabilities embodying mathematical aggregation functions like a *weighted mean*. The input of this task is a collection of weighted-pairs (v_i, w_i) consisting of a *value* (v_i) and a *weight* (w_i). The goal of this task is to obtain a single value for all the items referring to the same piece of information by aggregating its weighted values. Within the context of the Aggregation task-decomposer, this task is applied several times, one for each Item-info obtained during the Elaborate-item-infos task. Recall that an item-info encompasses all the information retrieved about a piece of information, like a bibliographical reference.

There are several methods that can be found in the literature for aggregating information based on numerical aggregation operators functions. We have included four methods in the library to perform this task: the *average*, the *weighted mean*, the *ordered weighted averaging* (OWA) and the *weighted ordered weighted averaging* (WOWA).

The *average* applies an arithmetic mean. The *weighted mean* is a lineal combination of values according to a vector of normalized weights. The weighted mean expects a number (say n) of weighted pairs to obtain a unique, aggregated value:

$$\frac{\sum_{i=1}^{n} v_i w_i}{\sum_{i=1}^{n} w_i}$$

The *ordered weighted averaging* (OWA) operator is, as the weighted mean, a lineal combination of the values according to a vector of weights. However, the difference is that OWA has a *weighting function* that establishes weights as a function of an ordering of the values to be aggregated.

First, using the OWA operator, the input values $\{v_i\}$ are (increasingly or decreasingly) ordered, giving an ordered collection of values $\{v_{\sigma(i)}\}$ where σ is the permutation performed. Then a *weighting function* Q assigns a weight to a value according to its position in that order:

$$\omega(i) = Q\left(\frac{i}{n}\right) - Q\left(\frac{i-1}{n}\right)$$

Finally, a lineal combination of the values with the assigned weights is computed:

$$\sum_{i=1}^{n} \omega_i v_{\sigma(i)}$$

OWA is parametric with respect to the weighting function: domain knowledge establishes the weighting function to be used in an application domain. Our approach is that the capability OWA imports a domain model or a particular application where the weighting function to be used is given as part of the domain knowledge.

The *weighted ordered weighted averaging* (WOWA) operator is a generalization of both the *weighted mean* and the OWA.

As in the OWA operator, the values $\{v_i\}$ are (increasingly or decreasingly) ordered. Then they are weighted according to their order using a weighting function, giving an ordered collection of values $\{v_{\sigma(i)}\}$ where σ is the permutation performed.

However, in addition to the weights derived from the order, WOWA uses the weights associated to the values in the weighted pairs. The combined weight ω is computed as follows:

$$\omega(i) = Q\left(\sum_{j\leq i} w_{\sigma(i)}\right) - Q\left(\sum_{j<i} w_{\sigma(i)}\right)$$

where $w_{\sigma(i)}$ denotes the permutation performed by the ordering upon the weights of the pairs (v_i, w_i).

Finally, a lineal combination of the values with the combined weights ω_i is computed as follows:

$$\sum_{i=1}^{n} \omega_i v_{\sigma(i)}$$

Let us remark here that the selection about which aggregation capability should be used is based on the preferences of the user as well as the type of information retrieved.

7.6 WIM domain knowledge

WIM uses domain knowledge from the fields of medicine and bibliographic data. This domain knowledge has been characterized by a collection of *domain-models*, which are specified by an ontology, plus a collection of properties and meta-knowledge, as described in §4.2.1 (Figure 4.12).

The main task for the WIM application is searching medical literature, and the utility criteria used to rank documents can be provided either by a medical thesaurus or by a collection of specific knowledge categories. In addition, in order to specify queries from an abstract, conceptual view, WIM needs bibliographic knowledge so as to specify queries to bibliographical search engines and handle bibliographical information. These are the three types of domain knowledge included in the WIM application, and thus there are three domain-models in WIM, one for each type of knowledge, namely the *MeSH* thesaurus, a categorization of *Evidence Based Medicine*, and a description of several bibliographical databases accessible through the Internet.

- MeSH is a general *medical thesaurus* that contains a huge collection of medical terms, and relations between terms. WIM agents include capabilities that can elaborate queries using terms from such a medical thesaurus. The

semantic relations contained in the thesaurus (synonyms, hyponyms and hypernyms, etc.) can be used by query elaboration capabilities to build new queries that are a generalization or a specialization of the original query.

- A collection of *source descriptions* is used to specify the search modes and filters allowed by each information source. In WIM information sources are accessed through agentified Web-based search engines, like Pubmed, which is used to access the Medline database, and Internet Grateful Med (IGM), which is used to access both Medline and HealthStar. A source description contains also a mapping between the bibliographical concepts used to generate queries at the conceptual level, and the concepts used by specific information sources. Source descriptions are used to customize queries for specific information sources.

- A collection of *categories* characterizing concepts from Evidence-based Medicine. This domain model characterizes a collection of *categories* in terms of keywords (terms) and filters that are useful to rank documents according to the EBM concepts. This knowledge is required by query elaboration capabilities such as the skill Query-expansion-with-category, to generate domain queries using the EBM category preferred by the user.

7.6.1 Evidence-Based Medicine

From the point of view of the Evidence-Based Medicine, it is very important to use the bibliographic references according to the quality of the evidence they rely on; hence, we have defined some categories expressing concepts about medical evidence quality in terms of a medical thesaurus and the defined ontology for bibliographic data, furthermore these concepts are weighted, allowing to weight the queries according to these "evidence quality" indicators.

Specifically, we have build 15 categories about EBM and some medical categories that are often required by medicine professionals using EBM. These categories belong to four different topics: *Evidence-Quality*, *Clinical Categories*, *Analysis* and *Evidence Integration*. See Figure 7.12 for a taxonomy of the medical categories defined in the EBM domain-model.

A category is defined as a structure of terms and filters associated to one topic, in which both filters and terms are weighted according to the strength of that association. Figure 7.7 shows the main concepts used to describe categories. For example *Guidelines* is a category that defines some filters to get only papers offering clinical advice based on good evidence quality. Table 7.6.1 shows the attribute-value definitions of the Guidelines category.

7.6.2 The MeSH thesaurus

A thesaurus is a book or a digital repository of synonyms, often including related and contrasting words and antonyms. There are general thesaurus concerning

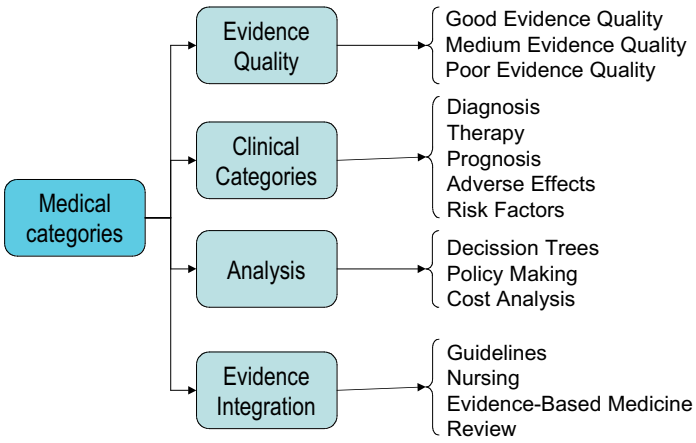

Figure 7.12: Categories on Evidence-based Medicine

Category	***Guidelines***	
Attribute	**Value**	**Weight**
Publication Type	Guidelines	1.0
Publication Type	Practice Guidelines	0.8
Publication Type	Guideline Adherence	0.6
Publication Type	Clinical Protocols	0.4

Table 7.1: Definition of the category *Guidelines*

MeSH Heading	Pneumonia
Tree Number	C08.381.677
Tree Number	C08.730.610
Scope Note	Inflammation of the lungs
Entry Term	Experimental Lung Inflammation
Entry Term	Lung Inflammation
Entry Term	Pneumonitis
Entry Term	Pulmonary Inflammation
Allowable Qualifiers	BL CF CI CL CN ...
Unique ID	D011014

Table 7.2: Definition of the term *Pneumonia* in Mesh

an entire language, but there are also specialized thesaurus containing concepts and vocabulary of a particular field, as of medicine or music.

The Medical Subject Headings (MeSH) is a controlled vocabulary produced by the National Library of Medicine and used for indexing, cataloging, and searching for biomedical and health-related information and documents. MeSH is the thesaurus used to index the bibliographic references stored in Medline, which is main database accessed by WIM information retrieval agents.

Table 7.6.2 shows an example of the information provided for the term *Pneumonia*, which is a MeSH heading. Each MeSH heading has a collection of synonyms specified as *Entry Terms* (e.g. Lung Inflammation, Pneumonitis and Pulmonary Inflammation), several *tree numbers* identifying the position of the term in the MeSH structure, and a collection of allowable modifiers (e.g. BL refers to *Blood*, used for the presence or analysis of substances in the blood; also for examination of, or changes in, the blood in disease states).

However, WIM capabilities can not use the MeSH information straightforwardly; instead, WIM capabilities are based on the ISA Ontology, which defines a term as having parent terms, children terms and other related terms specified as term-correlations. Therefore, we have implemented a mapping from MeSH headings to WIM terms so as to allow WIM capabilities to work with them. Entry terms become term-correlations with a correlation equal to one, which the maximum, because entry terms are synonyms of the main term (a MeSH heading). The *parents* (hyponyms) and the *children* (hypernyms) of a term are obtained from the MeSH structure using the tree number. For instance, *Pneumonia* has *Lung Diseases* as parent, and *Bronchopneumonia*, *Pleuropneumonia*, *Aspiration Pneumonia*, *Bacterial Pneumonia* and others as children. Although it is possible to download MeSH from the Web to use it locally, we have decided to access MeSH directly through the Web-based MeSH Browser, thus we have built a Wrapper to retrieve MeSH information from the Web and transform it into WIM terms. An example of a term is the following:

```
(define (term :id Pneumonia)
  (name "Pneumonia")
  (term-correlations (define (Term-Correlation)
                        (term Lung Inflammation)
                        (weight 1))
                     (define (Term-Correlation)
                        (term Pneumonitis)
                        (weight 1))
                     (define (Term-Correlation)
                        (term Lung Inflammation)
                        (weight 1))
                     (define (Term-Correlation)
                        (term Pulmonary Inflammation)
                        (weight 1))))
  (parent Lung Diseases)
  (children Bronchopneumonia
            Pleuropneumonia
            Aspiration
            Pneumonia
            Bacterial
            Pneumonia))
```

The idea of using term-correlations instead of synonyms is to allow more complex relations between words, like those provided by a Latent Semantic Analysis.

7.6.3 Medical sources

In order to facilitate the integration of multiple information sources, we have decoupled the knowledge on information sources from the query customization and retrieval capabilities. The idea of WIM is to provide a mediation service between the user putting a query, and multiple information sources. Such a mediation service must offer the user a single interface to many search engines, and aggregate information coming from heterogeneous sources. This type of mediation is often called Intelligent Information Integration (I3). As a result of reviewing many Information Integration systems (e.g. the Information Manifold [Kirk et al., 1995, Levy et al., 1996], TSIMMIS [Chawathe et al., 1994], Infomaster [Genesereth et al., 1997], Infosleuth [Nodine et al., 1999], and SIMS [Arens et al., 1993, Knoblock et al., 1994]), we concluded the importance of using a *lingua franca* to specify user level queries, and mapping schemas to transform concepts specified in the lingua franca into concepts used by specific information sources. In WIM, the queries expressed in the lingua franca are called *domain queries*, and the source-specific queries are called *source queries*.

On the one hand, we have elaborated a bibliographic-data ontology (Figure 7.13) encompassing the usual concepts found in the most popular bibliographical databases in medicine: Medline and Healthstar. Some of these concepts are the

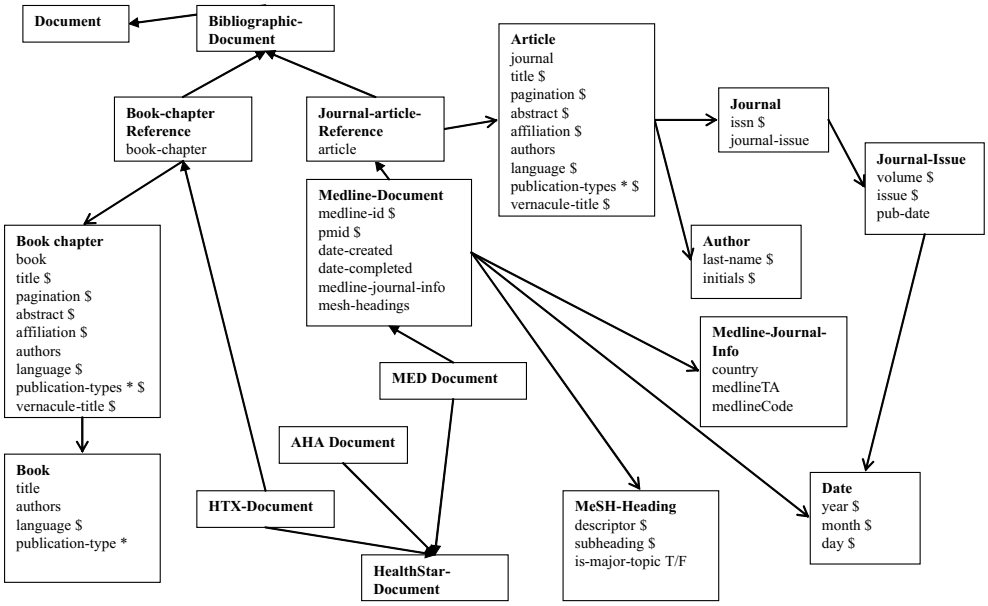

Figure 7.13: *Bibliographic-data* ontology

following:

- author name,
- publication type (e.g. journal, book chapter, proceedings, etc.),
- language,
- affiliation of the authors,
- begin year (lower-limit for publication date),
- end year (upper-limit for publication date),
- substance (drugs and pharmacologic terms),
- study groups (for statistically-based studies)
- age groups (for statistically-based studies)

On the other hand we have build an ontology to describe information sources . Recall that information sources in WIM are not databases as such, instead they are Web based search engines used to access the content within a database; for example, Pubmed is used to access Medline, and Internet Grateful Med is used to access Healthstar. There are two types of elements used to specify queries in Web-based search engines concerning bibliographical databases (Figure 7.8): search modes and search filters. Search attributes refer to different search modes

allowed by a search engine (e.g. searching a keyword only at the title, the abstract or both); moreover, we distinguish between the basic or default search mode (*basic attribute*), and other search modes supported by a search engine (*search-attributes*). Filters are used to constrain the search, using bibliographic concepts from the subsect of bibliographic concepts selected.

In WIM search modes are not specified by the user, they are incorporated into a query by the query customization capabilities in order to assess the results (§7.2) of a query. Therefore, only the filters require a mapping schema definition to translate the filters specified by the user (domain attributes) to equivalent filters expressed in a source-specific vocabulary (source attributes). An example of a source description is included in §7.7.2.

7.7 Exemplification of the WIM library

We have already introduced the domain knowledge used in the WIM application: the MeSH thesaurus, the EBM categories and the information sources encompassing bibliographical references about medicine. In this section we will show by means of an example how the domain knowledge is used to generate queries from a user consultation, and how the results are aggregated and ranked with respect to the relevance and utility of the user consultation. Section 7.7 shows how WIM configures a team for a specific problem, from the problem specification stage to the solution of the problem by a team of agents.

7.7.1 Query Elaboration using EBM

Suppose the user wants to find information about *guidelines* on the use of *levofloxacin* in the treatment of *pneumonia*. Therefore, the selected keywords should be the name of the substance (levofloxacin), the disease (pneumonia), and the type of information searched (guidelines). In addition, the user decides to limit the search to articles published after 1980, thereby he specifies a filter for the attribute *Begin Year* with a value of 1980. The consultation is equivalent to the the following query

```
query.terms: levofloxacin, pneumonia, guidelines
query.filters: (Begin Year, 1980)
```

The agent applying the Query-expansion-with-categories skill takes the knowledge on the Guidelines category (see Table 7.6.1) from the EBM knowledge base. That agent find that the Guidelines category is related (with a specific correlation strength) to the following terms: “practice-guidelines” (with strength of 0.8), “guideline-adherence” (with strength 0.6), and “clinical-protocols” (with strength 0.4).

The system uses term correlations to generate new *query models* where the original keywords are replaced by the correlated terms. The output of the capability is a collection of query models with weights corresponding to the correlation strength of the term being used. Only one term is replaced by a correlated

term at each output query. In the example, the generated query models have a domain query whose terms are the same terms specified as keywords in the user's query except for "guidelines", that is replaced by one of their correlated terms —practice-guidelines, guideline-adherence and clinical-protocols.

```
Query model 1 Query.terms:
levofloxacin pneumonia guidelines Query.filters: (Begin Year,
1980) Weight: 1

Query model 2 Query.terms: levofloxacin pneumonia
practice-guidelines Query.filters: (Begin Year, 1980) Weight: 0.8

Query model 3 Query.terms: levofloxacin pneumonia
guideline-adherence Query.filters: (Begin Year, 1980) Weight: 0.6

Query model 4 Query.terms: levofloxacin pneumonia
clinical-protocols Query.filters: (Begin Year, 1980) Weight: 0.4
```

7.7.2 Basic Query Customization

Once the Elaborate-query task is finished, and before the start of the Customize-query task, some information sources are selected from a set of allowable sources. For simplicity, let's assume the two sources have been selected: *Pubmed* and *HealthStar*. Then, the capability selected to solve the Customize-query has to adapt the domain queries obtained by the Query-expansion-with-categories capability to those two information sources.

The Query-customization capability expands a query expressed in a source independent way (a domain query) into a collection of queries in terms of a particular information source (source queries), using the search modes and filters allowed by each source. The properties of this knowledgeare characterized by a domain-model called Source-Descriptions. The source-description for the Pubmed information source (a Web-based retrieval engine to the Medline database) is included below as an example.

```
(define (source :id Pubmed)
 (name "PubMed")
 (weight 1)
 (search-attributes
                      (define (Attribute-Weighting)
                        (attribute "MAJR")
                        (weight 1))
                      (define (Attribute-Weighting)
                        (attribute "MH:NOEXP")
                        (weight 0.8))
                      (define (Attribute-Weighting)
                        (attribute "MH")
```

```
                        (weight 0.6))
                      (define (Attribute-Weighting)
                        (attribute "TI")
                        (weight 0.5))
                      (define (Attribute-Weighting)
                        (attribute "TW")
                        (weight 0.4)))
  (basic-attribute (define (Attribute-Weighting)
                        (attribute "ALL")
                        (weight 0.2)))
  (filter-attributes
                      (define (Attribute-Translation)
                        (domain-attribute "Author Name")
                        (source-attribute "AU"))
                      (define (Attribute-Translation)
                        (domain-attribute "Publication Type")
                        (source-attribute "PT"))
                      (define (Attribute-Translation)
                        (domain-attribute "Language")
                        (source-attribute "LA"))
                      (define (Attribute-Translation)
                        (domain-attribute "Affiliation")
                        (source-attribute "AF"))
                      (define (Attribute-Translation)
                        (domain-attribute "Journal")
                        (source-attribute "JO"))
                      (define (Attribute-Translation)
                        (domain-attribute "Begin Year")
                        (source-attribute "MINDATE"))
                      (define (Attribute-Translation)
                        (domain-attribute "End Year")
                        (source-attribute "MAXDATE"))))
```

The Query-customization capability (a skill) performs two kind of transformations for each pair consisting of a query and a source:

- *from domain filters to source filters*: the query filters, written in terms of the domain terminology (bibliographic data) are rewritten in terms of the source terminology, according to the attribute-translations in the source domain-model; and
- *from keywords to search-attributes*: each search term is transformed into a filter where the value is the term and the attribute is one of the search-attributes allowed by the source. These filters are called *t-filters* in the ISA-Ontology, though they are objects of sort Filter.

A new query model is created using the *basic-attribute* to build the *t-filters*. The *filters* are translated using the translation rules defined in the Source De-

scriptions. For example, for the Query Model 1 and the source Pubmed, the following transformation is applied.

```
Query Model 1 before the customization:
    Query.terms: levofloxacin pneumonia guidelines
    Query.filters: (Begin Year, 1980)
    Weight: 1

Query Model 1 after the customization:
    Query.t-filters: (ALL, levofloxacin), (ALL, pneumonia),
                     (ALL, clinical-protocols)
    Query.filters: (MINDATE, 1980)
    Source: PubMed-Medline weight: 0.2
```

Note that all the *t-filters* use the basic attribute (called "ALL") and the attribute "Begin Year" is replaced by MINDATE.

A new query model is created for each *term* and each *search-attribute* in the source domain-model. The following query models are obtained when using the PubMed *search-attributes* (TW, TI, MH, MH:NOEXP and MJR) in place of the basic-attribute (ALL), for the first term.

```
Query model 1.2.a
    query.t-filters: (TW, levofloxacin), (ALL, pneumonia),
                     (ALL, guidelines)
    query.filters: (MINDATE, 1980)
    weight: 0.4

Query model 1.3.a
    query.t-filters: (TI, levofloxacin), (ALL, pneumonia),
                     (ALL, guidelines)
    query.filters: (MINDATE, 1980)
    weight: 0.5

Query model 1.4.a
    query.t-filters: (MH, levofloxacin), (ALL, pneumonia),
                     (ALL, guidelines)
    query.filters: (MINDATE, 1980)
    weight: 0.6

Query model 1.5.a
    query.t-filters: (MHNOEXP, levofloxacin), (ALL, pneumonia),
                     (ALL, guidelines)
    query.filters: (MINDATE, 1980)
    weight: 0.8
```

```
Query model 1.6.a
    query.t-filters: (MAJR, levofloxacin), (ALL, pneumonia),
                     (ALL, guidelines)
    query.filters: (MINDATE, 1980)
    weight: 1
```

In the example above each *query model* is identified by two numbers and a letter separated by dots: QM x.y.z identifies the y^th *query-model* (containing a source query) obtained for the x^{th} *query-model* (containing a domain query) and source z (a for Pubmed and b for IGM-HealthStar). The weights of the resulting *query-models* are calculated multiplying the weight of the input *query-model* by the weight of the *search-attribute* used, or multiplying by the weight of the *basic-attribute* if no *search-attribute* is used (e.g. QM 1.1.a).

Modifying the *search-attribute* for the second or third *term* and keeping the other terms associated to the *basic-attribute* results in 10 new query models, 5 per term (QM 1.7a – 1.11a and 1.12a – 1.16a).

```
Query model 1.7.a
    query.t-filters: (ALL, levofloxacin), (TW, pneumonia),
                     (ALL, guidelines)
    query.filters: (MINDATE, 1980)
    weight: 0.4
    ...

Query model 1.11.a
    query.t-filters: (ALL, levofloxacin), (MAJR, pneumonia),
                     (ALL, guidelines)
    query.filters: (MINDATE, 1980)
    weight: 1
    ...

Query model 1.16.a
    query.t-filters: (ALL, levofloxacin), (ALL, pneumonia),
                     (MAJR, guidelines)
    query.filters: (MINDATE, 1980)
    weight: 1
```

At the end, each domain-query has originated 16 source-queries when customized for PubMed; therefore, since the user consultation has produced 4 *domain-queries* (QM1a, QM4), a total of 4*16 = 84 source-queries are obtained for PubMed. The same *domain-queries* are customized for HealthStar, which has only *search-attribute* different from the *basic-attribute*, therefore only 4 source queries are generated for each domain-query, summing 16 queries in total.

Finally, the 84 Pubmed queries and the 16 Healthstar queries are sent to the wrappers to each information source, which perform the final transformation of the queries to fit the specific interface of each information source.

Let us briefly explain the operation of a wrapper, for instance the Pubmed-wrapper. The wrapper concatenates at least three strings: the server URL, a path to the source location in the server and the query itself, according to the source query format. For example, for the Pubmed-wrapper and the Query Model 1.6.a:

```
Server URL: "http://www.ncbi.nlm.nih.gov/"
Path: "entrez/query.fcgi?"
query.t-filters: (MAJR, levofloxacin), (ALL, pneumonia),
                                       (ALL, guidelines)
query.filters: (MINDATE, 1980)
```

The resulting URL is

```
entrez/query.fcgi?CMD=search\& DB=PubMed\&term=
levofloxacin[MAJR]+AND+pneumonia[ALL]+AND+
guidelines[ALL]+AND+1980[MINDATE]
```

The wrapper also deals with the problem of the format in which results are retrieved, the number of items to retrieve and other technical issues.

7.7.3 Aggregation

When a weight is assigned to a query after applying some transformation, this is expressing the relative importance or representativity of that query with respect to the original one. The meaning of a weight assigned to a query is logically inherited by the documents or items retrieved for that query, thus we can say that the weights associated to the items retrieved represent the membership of those elements to the topic requested by the user. Aggregation operators for numeric values can be used here. The most widely used operators are the *arithmetic mean* and the *weighted mean*, but there is a family of aggregation operators, including fuzzy measures; WIM implements several operators reviewed in [Torra, 1996].

Let's see an example. First, suppose that Pubmed has been queried using four queries with the following weights: 1, 0.8, 0.6, and 0.4. Suppose that one of the retrieved items appears in queries 1 and 2, but not in 3 and 4. As PubMed does not rank documents, WIM assigns by default a maximum score of one to both occurrences of the same item. When doing the aggregation, absence of items are also taken into account; each absence is represented like a "presence" with a value equals to 0, as showed below.

Query	w_i	$\bar{w}_i$	$score_i$
Q_1	1,0	0,36	1
Q_2	0,8	0,29	1
Q_3	0,6	0,21	0
Q_4	0,4	0,14	0

Before any aggregation, the weights of the queries have to normalize, because aggregation operators assume that $w_i \in [0,1]$ and $\sum_{\forall i} w_i = 1$. We denote this

normalized weight as $\bar{w}$. We can apply for instance a weighted-mean operator, defined as $WM(Q_1, \ldots, Q_n) = \sum_{i=1\ldots n} \bar{w}_i \cdot score_i$, and the aggregate value is 0,65.

After finishing the aggregation the user gets the results of his consultation; some examples of the *scored-items* obtained are shown below:

```
Scored item 1
    identifier: PMID10743984
    title: Economic assessment of the community-acquired pneumonia
    intervention trial employing levofloxacin.
    score: 0.187

Scored item 2
    identifier: PMID10683053
    title: A controlled trial of a critical pathway for treatment
    of community-acquired pneumonia
    score: 0.172

Scored item 1
    identifier: PMID11557471
    title: Pharmacodynamics of fluoroquinolones against
    Streptococcus pneumoniae in patients with community-acquired
    respiratory tract infections
    score: 0.062
```

7.8 Experimental results

We have developed two applications where the query weighting approach has been used: MELISA and WIM.

MELISA [Abasolo and Gómez, 2000] is a single agent to look for medical literature, which is based on the use of ontologies to separate the domain knowledge from source descriptions. The notions of query weighting and exploiting the filtering capabilities of existing search-engines are already used here, but only one search-engine is included. Although the aggregation procedures were not systematized yet, the system demonstrated that the query weighting and aggregating framework provides an accurate and flexible approach to rank documents. MELISA has proven the flexibility of this framework to apply utility-criteria to rank documents. In particular, a collection of knowledge categories has been used to rank medical references according to the quality of the evidence they are based on, which is called Evidence Based Medicine [Feinstein and Horwitz, 1997]. The query weighting process is applied twice, one at the domain level and the other at the source level. Query weighting at the domain level is carried over using knowledge categories; each category is a medical topic described as a collection of weighted elements. The elements used to describe medical categories are medical terms, used as keywords, and other search attributes used to look for

bibliographic references, like publication type, related terms, and so on. Query expansion at the source level is achieved by using the search modes described in §7.7.2 for the Pubmed source.

We have compared the results of MELISA (using PubMed as the search engine) against those results obtained with the PubMed for some user queries. An expert has proposed us 5 cases to look for medical references based on his everyday work. Every case has been translated into a query suitable for both PubMed and MELISA. The results have been scored by two different evaluators, instructing them to score the references according to their degree of relevance for the corresponding query and the quality of the evidence. The first 40 references retrieved for each query have been evaluated by the experts. An ordinal scale with three levels has been used to score references: 2 for references that satisfy the user's need, 1 if the reference is simply related, 0 if it is not relevant, and "?" if the expert has no enough information (the expert has only the title and abstract of the documents) about a bibliographic reference to asses its relevance.

Table 7.3 sums up the results of the assessment procedure for the relevance. As you can see in the table, the results obtained by MELISA have been evaluated as being more relevant than those obtained by querying Pubmed directly. Specifically, there are more bibliographic references considered relevant for the user's need, and fewer references ranked irrelevant or without enough information to be evaluated.

Relevance	PubMed	MELISA
2	38%	48%
1	16%	17%
0	13%	9%
?	34%	27%

Table 7.3: Assessment of relevance

The utility of the proposed framework to develop and use new ranking criteria has also been tested. Some of the implemented medical categories are specifically designed to represent notions about evidence quality in medical references. Using these categories to enrich queries during the query expansion has demonstrated that the resulting references are well ranked according to the quality of the evidence. The testers have been instructed to classify references into three levels: good, medium and poor evidence. The symbol "?" is used to note references without enough information to be evaluated according to the quality of the evidence.

Table 7.4 sums up the results of the assessment procedure concerning the quality of the evidence. As you can see in the table, results are not so easy to interpret as for the relevance. Notice that there are a greater percentage of items assessed as being based on a good evidence quality (20% to 7%), though surprisingly there are also more items that have been assessed as backed with a poor evidence quality (9% to 2%). Another datum to be remarked is the very high percentage of items with not enough information. The point is that we

are working with the bibliographic references only, and not with the documents (mainly articles) being referenced, thus usually there is no enough information to assess the quality of the evidence a referenced document is based on.

Evidence	PubMed	MELISA
Good	7%	20%
Medium	2%	3%
Poor	2%	9%
?	89%	69%

Table 7.4: Assessment of evidence quality

To sum up, MELISA has proven the utility of the information search approach adopted in WIM to rank documents according to utility criteria like EBM quality: in our experiments, MELISA has received a stronger approval assessment from expert user than PubMed.

However, we expect WIM to obtain similar or even better results than MELISA, since WIM extends the core functionality implemented in MELISA. The reason is that MELISA implements a fixed combination of query elaboration, customization and aggregation methods, whilst WIM allows for a multiplicity of tasks and methods that can be combined in different ways through the Knowledge Configuration process. Moreover, WIM is configured on-demand according to problem requirements, thus the resulting agent team is tailored to the problem at hand to better fit the needs and preferences of the user. The present evaluation, assesses just the core functionality of the new information search techniques implemented in WIM's library.

7.9 Example of the Cooperative Problem Solving process in WIM

This section illustrates the use of the ORCAS framework in the WIM application to carry out an information search task by customizing a team of problem solving agents on-demand, according to the ORCAS model of the Cooperative Problem Solving process. We show a complete example on how WIM operates, following the different stages of the CPS process as implemented in the ORCAS institution, which encompasses the following processes: Registering of capabilities, Brokering (Knowledge Configuration), Team Formation and Teamwork. In addition, the Problem Specification process is included to show the way a problem is specified by a user or an external agent.

The WIM example is illustrated with some snapshots from the Agent World, a 3D World environment that represents the agent communication and agent flow between scenes in the e-institution (external view), together with information on agents state (internal view). The Agent World external view shows an e-Institution as a set of rooms representing the scenes (big rectangles), and corridors representing transitions between scenes (little rectangles connecting one

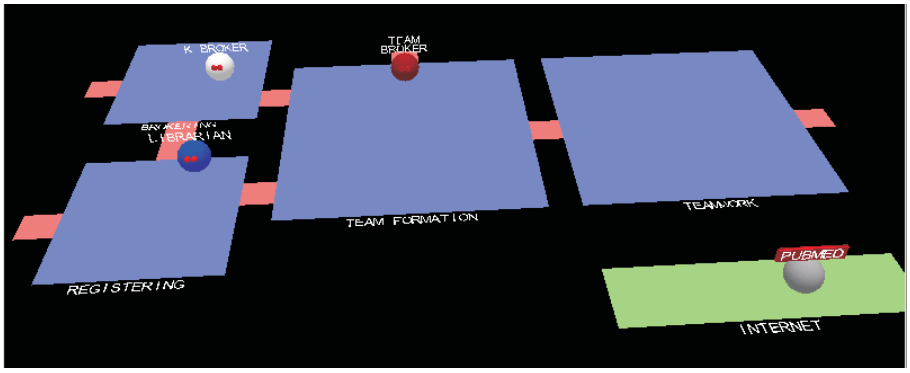

Figure 7.14: Overview of the ORCAS e-Institution

scene to another). Agents are represented as eyed-balls staying at a room or moving between rooms through transitions, and the messages they send are represented as little balls colored according to the type of message (e.g. request, inform, etc.).

Image 7.14 shows a zoomed out view of the ORCAS e-Institution in the Agent World tool. One can see the main scenes of the ORCAS e-Institution and the allowed transitions between scenes, together with some institutional agents waiting for other agents. At the Registering scene there is a Librarian agent waiting for new Problem Solving Agents (PSAs) willing to join the institution; at the Brokering scene there is a Knowledge-Broker waiting for a Personal Assistant (PA) having a problem to be solved; and at the Team Formation scene there is a Team-Broker waiting for both a PA willing to form a team of agents, and a set of PSAs providing their capabilities.

7.9.1 Registering capabilities

In order to become available to other agents, PSAs must register their capabilities to a Librarian agent, according to the specification of the Registering scene. Capabilities are registered using the ORCAS Knowledge Modelling Ontology and Feature Terms as the Object Language.

Figure 7.15 shows a screenshot of the Registering scene. After registering their capabilities to the Librarian, the PSAs move to the Team Formation scene to wait for team-role proposals to joining a team. At the moment captured, the PSA Marteen has just finished the Registering scene and is moving to the Team Formation scene.

The Registering scene is instantiated once for each new PSA entering the institution, but the Agent World monitoring tool shows only one scene that is a synthesis of all the instances of the scene occurring simultaneously; therefore, several PSAs can be shown in the Registering scene at a time.

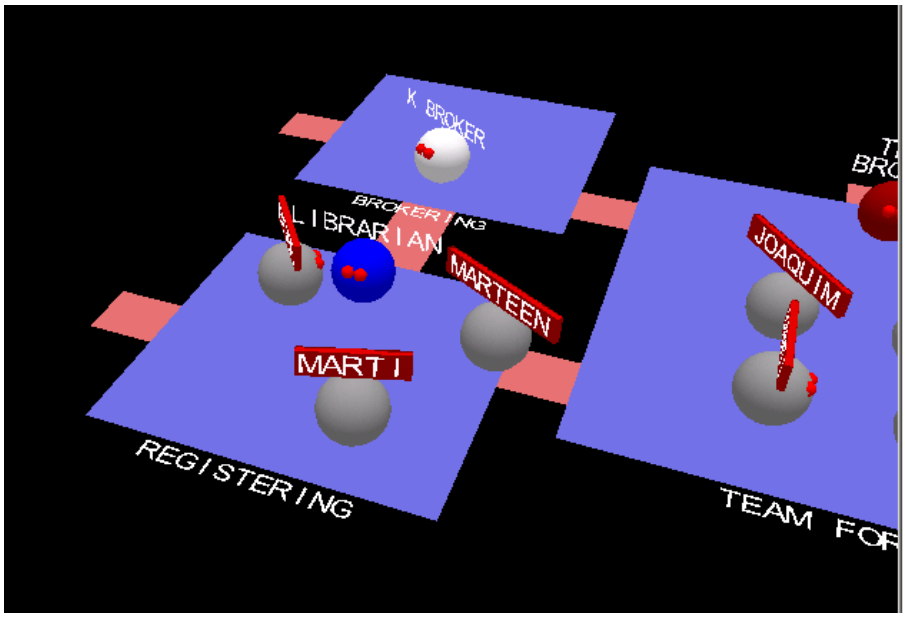

Figure 7.15: Screenshot of the Registering scene

The Librarian keeps an updated register of the capabilities registered by the PSAs that have joined the system. Figure 7.16 shows a screenshot of the Librarian internal state displaying the record of capabilities registered in the institution by WIM PSAs.

7.9.2 Problem specification

The WIM application can process requests coming from different sources: from a windows-like client, from Web pages, and from external agents using either the NOOS Agent Communication Language (ACL) or the FIPA-ACL (§7.3 describes the overall architecture). We will show the way a problem is specified by the end user and the way it is encoded in agent terms.

A problem is specified by input data and problem requirements, as described in §4.4.2. The input data for an information search task in WIM is a consultation, composed of a set of keywords and filters plus a set of information sources and optionally a knowledge category. We are going to show first how a problem is specified using the WIM panel, and the way a problem is specified using the Agent Communication Language afterwards.

Figure 7.17 shows the consultation screen where the user specifies a consultation, including just a portion of all the filters allowed by the WIM application (see §F.3). In the example, the user has introduced three keywords (ofloxacin, pneumonia and guidelines) and a filter (from year 1980). The user has a limited control over the type of task to be solved, either the task Information Search

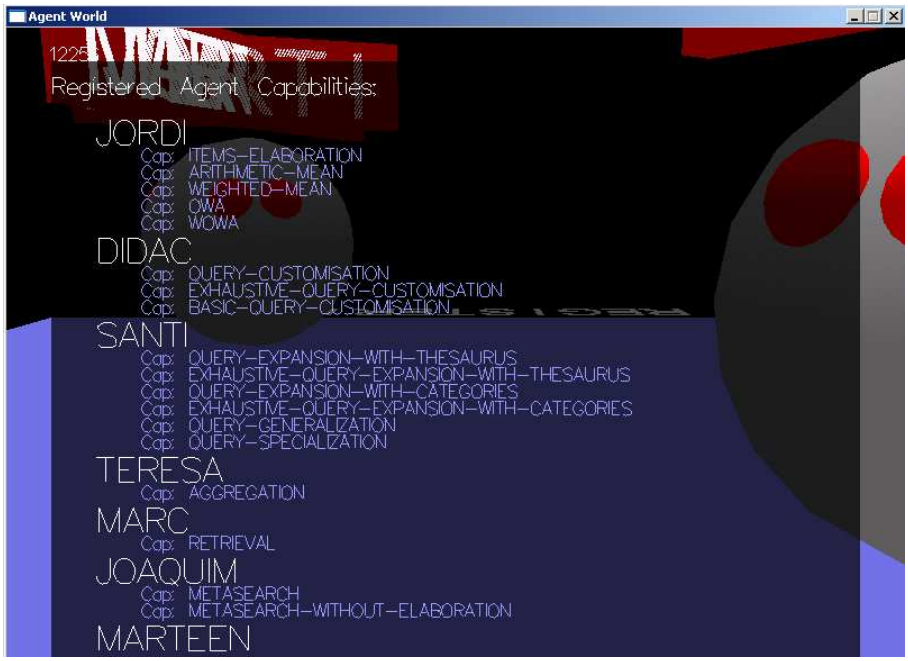

Figure 7.16: Example of the Librarian internal state

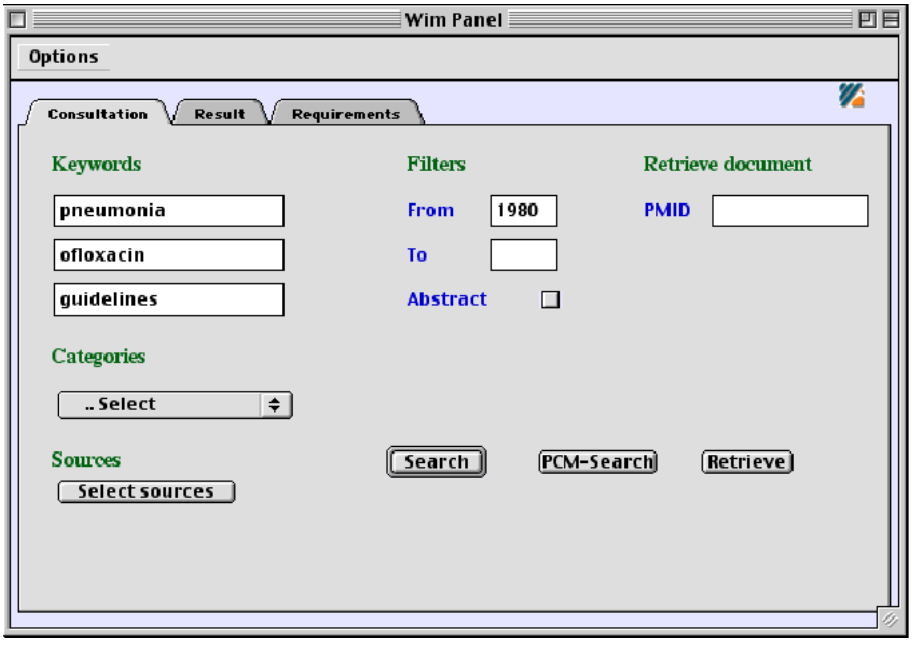

Figure 7.17: Consultation example

(buttons labelled `Search` and `PCM-Search`) and the task Retrieve-full-reference (button labelled `Retrieve` in the figure). Furthermore, the user can decide whether to apply the task-decomposer Metasearch or the task-decomposer PCM-Search to solve the Information Search task; which is discriminated by pushing either the button labelled `Search` or the one labelled `PCM-Search`.

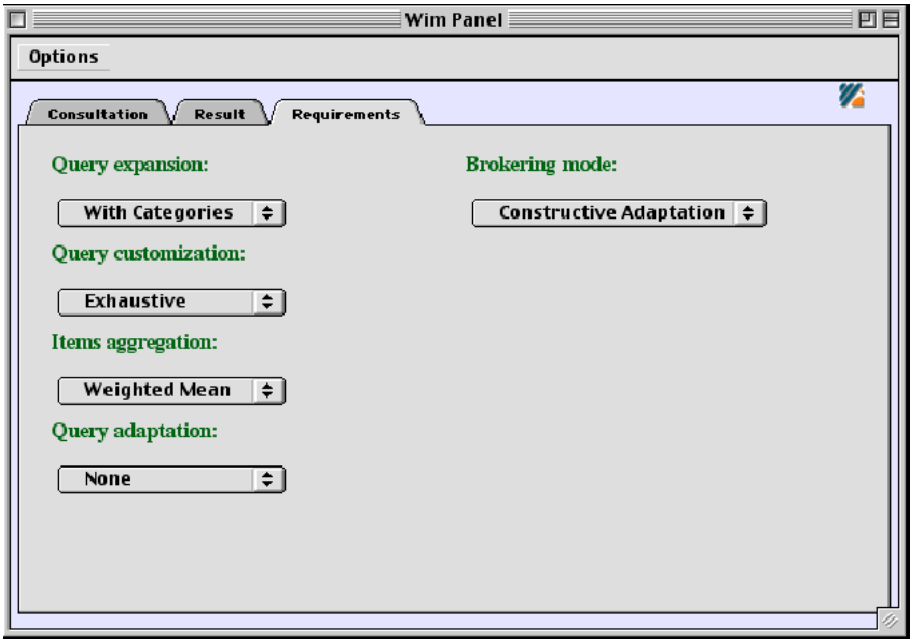

Figure 7.18: Consultation example

Figure 7.18 shows the screen where the user specifies the problem requirements. That screen is an interface where the user chooses some among a limited number of requirements organized according to the tasks they are related with (as a curiosity, compare it with the interface showed in 4.23, which is addressed to the knowledge/software engineer and offers all the requirements available in the library following a plain, quite raw style).

In our example, the user has selected some options corresponding to the following postconditions:

- Elaborate-query-with-thesaurus
- Non-exhaustive-customization
- Aggregate-with-arithmetic-mean

Each option corresponds to a postcondition added to the problem requirements to discriminate among alternative capabilities for the same task. Other requirements are added automatically by the GUI depending on the button pushed

by the user. For instance, if the user pushes the button labelled `PCM-Search`, two extra postconditions will be added to the requirements to ensure that the capability PCM-Search will be selected during the Knowledge Configuration process instead of Metasearch:

- Satisfy-consult
- Asses-search-results

Moreover, the options screen allows the user to select either the *Search and Subsume* or the *Constructive Adaptation* strategy (see §4.4.4) for the Knowledge Configuration process.

Once a problem has been specified using whatever means, it should be encoded in agent understandable terms, using either the NOOS ACL or the FIPA-ACL. We show below a FIPA-ACL specification of a message requesting the WIM PA to perform a Cooperative Problem Solving process according to the requirements stated above, assuming the user has pushed the button labelled `PCM-Search`.

The request message includes both the problem requirements and the problem data corresponding to the former example, encoded in XML within the content of the message. Notice that the XML encoded data is embedded within a FIPA-SL0 envelop defining an "action". See Appendix F for a more detailed account of the ORCAS services.

```
(request
    :sender     (agent-identifier :name uva-agent@a1136.fmg-uva-nl:1099/JADE)
    :receiver   (set (agent-identifier :name \wim-PA@wim.iiia.csic.es:7778/NOOS))
    :reply-with     cps-request18236
    :encoding   String
    :language   FIPA-SL0
    :ontology   WIM-Ontology
    :protocol   FIPA-request
    :conversation-id    cps18236
    :content
       (action
        (agent-identifier :name uva-agent@a1136.fmg-uva-nl:1099/JADE)
        (Cooperative-Problem-Solving
           :problem-requirements
              (Problem-Requirements
                 :encoding  <xml? version="1.0" encoding ="ISO-8859-1"?>
             :value
                <problem-requirements>
                <task-name>PCM-Search</task-name>
                <postconditions>
                   <formula>Elaborate-With-Thesaurus</formula>
                   <formula>Non-Exhaustive-Customization</formula>
                   <formula>Aggregate-With-Arithmetic-Mean</formula>
                   <formula>Satisfy-Consult</formula>
                   <formula>Assess-Search-Results</formula>
                </postconditions>
                <input-roles>
                   <signature-element>Query-Model</signature-element>
                </input-roles>
                <domain-models>
                   <domain-model>Medical-Sources</domain-model>
                   <domain-model>MeSH</domain-model>
                   <domain-model>EBM</domain-model>
                </domain-models>
```

```
    </problem-requirements>)
:problem-data
   (User-Consult
      :encoding  <xml? version="1.0" encoding ="ISO-8859-1"?>
  :value
     <user-consult>
     <query>
     <keywords>
        <keyword>Ofloxacin</keyword>
        <keyword>Pneumonia</keyword>
        <keyword>Guidelines</keyword>
     </keywords>
     <filters>
         <filter>
             <attribute>Begin-year</attribute>
             <value>1980</value>
         </filter>
     </filters>
     </query>
     <sources>
        <source>Pubmed</source>
     </sources>
     </user-consult>))))
```

The application task is Information-Search, and there are five postconditions in total, the first three ones specified by the user, and the other two added by the GUI, as explained above. Notice there are three domain-models added to the problem specification: Medical-sources, that provides a knowledge-role of sort Information-Sources, MeSH, that provides a knowledge-role of sort Thesaurus, and EBM, that provides a knowledge-role of sort Categories. Recall that the knowledge to be used by the selected capabilities must conform to the properties of the domain-models specified within the problem requirements (see §4.4.2 or §4.4.5). Therefore, all the domain-models available in WIM are included in the request; only the capabilities which knowledge-requirements are satisfied by one of these domain-models are potential candidates to be selected during the Knowledge Configuration process.

The Personal Assistant is able to parse messages specified in FIPA-ACL, extract the embedded data and translate them into the NOOS ACL before communicating with other agents to carry out the requested action, performing a Cooperative Problem Solving process throughout. To do that, the PA must participate in three different scenes within the ORCAS e-Institution: first, the PA goes to the Brokering scene to request the Knowledge-Broker for a task-configuration satisfying the problem requirements; second, the PA moves to the Team Formation scene and request the Team-Broker to form a team according to the just obtained task-configuration; and third, the PA moves to the Teamwork scene and solicits the team-leader to solve the problem using the specified problem data.

7.9.3 Knowledge Configuration

The Knowledge Configuration process has the goal of finding a configuration of agent capabilities satisfying the requirements of a problem to be solved. These requirements are specified as signatures (inputs, outputs and domain-models)

and formulae (preconditions and postconditions), represented with the same language that the agent capabilities, thus allowing to match the problem specification against the registered agent capabilities.

Recall that the Knowledge Configuration process is performed by the Knowledge-Broker (K-Broker) at the Brokering scene. The Brokering scene (§6.5.2) starts with the PA sending a problem specification to the K-Broker to request for a task-configuration satisfying that specification. Figure 7.19 shows the Brokering scene when the PA has just send the *request* message (represented by a small ball in the figure) to the K-Broker.

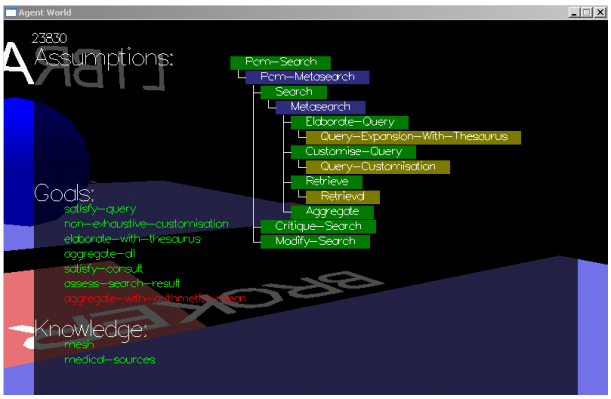

Figure 7.19: Screenshot of a Brokering scene

Following, the K-Broker asks the Librarian to get an up-to-date version of the tasks and capabilities available in the system, and after that, the K-Broker starts a Knowledge Configuration process over those specifications.

Figure 7.20 shows a screenshot of the K-Broker internal state some moment before finding a task-configuration. The left side of the screen shows the assumptions and postconditions (called goals), both the already satisfied ones (green/light gray) and the ones to be yet satisfied (red/dark gray), plus the selected domain models (knowledge). The right side of the screen shows the ongoing task-configuration.

Notice that there are more postconditions than those specified by the problem requirements, like SATISFY-QUERY and AGGREGATE-ALL. These postconditions have been introduced to the search state as a result of the tasks having been added during the Knowledge Configuration process by a task-decomposer. For instance, the postcondition SATISFY-QUERY has been added by the task Retrieve.

The task Information-Search has been bound to the task-decomposer PCM-Metasearch, which introduces three new tasks: P-Search, C-Search and M-Search. The task P-Search is bound to the capability Metasearch, which introduces four subtasks: Elaborate-query, Customize-query, Retrieve and Aggregate.

The task Elaborate-query is bound to the skill Query-expansion-with-thesaurus, because this is the only capability satisfying the user specified postcondition

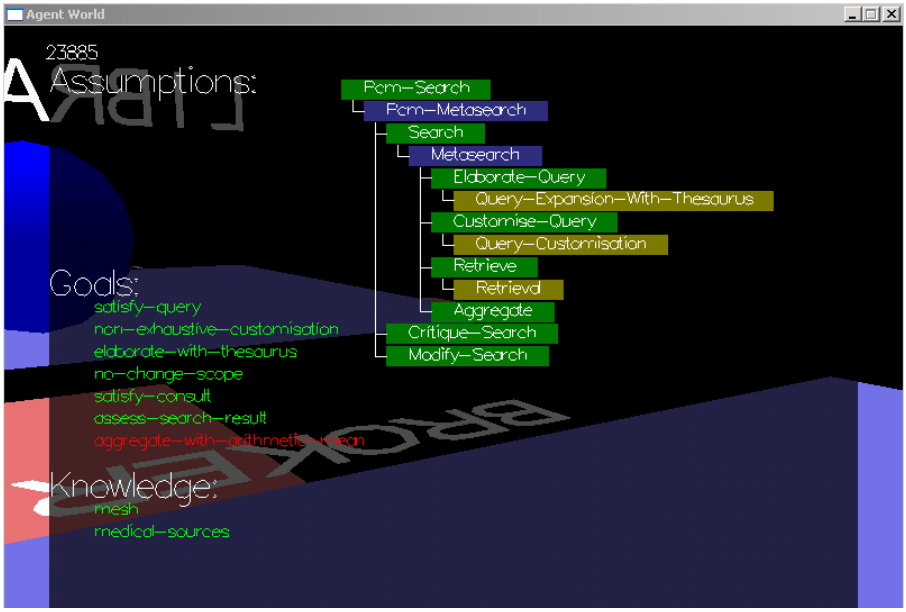

Figure 7.20: Example of the K-Broker internal state

ELABORATE-QUERY-WITH-THESAURUS. Customize-query is bound to Query-customization, due to the postcondition NON-EXHAUSTIVE-CUSTOMIZATION. Retrieve is bound to Retrieval, which is the only capability matching that task. However, Aggregate is not bound to a capability yet, and in accordance with it, the postcondition AGGREGATE-WITH-ARITHMETIC-MEAN has yet to be satisfied.

Figure 7.21 shows the final task-configuration for the current example. Notice that the task M-Search is not being configured because it depends on the result of the task P-Search, or more specifically, it depends on the result of the C-Search task, which has the purpose of assessing the result of P-Search. This example corresponds to the *Delayed Configuration* mode described in §5.7.1.

Finally, the K-Broker will send the PA an "inform" message containing the task-configuration in Figure 7.21, and after that the PA will move to the Team Formation scene where it will request the T-Broker to form a new team according to that task-configuration.

7.9.4 Team-formation

Team Formation is decomposed into three different activities: task allocation, team selection, and team instruction (§6.5.3). All these activities are coordinated by an agent playing the Team-Broker (T-Broker) role, following an auction-like interaction protocol.

Figure 7.22 captures the moment when the Team-Broker is sending team-role proposals to the available PSAs (e.g. Santi, Didac, Jordi), which will answer

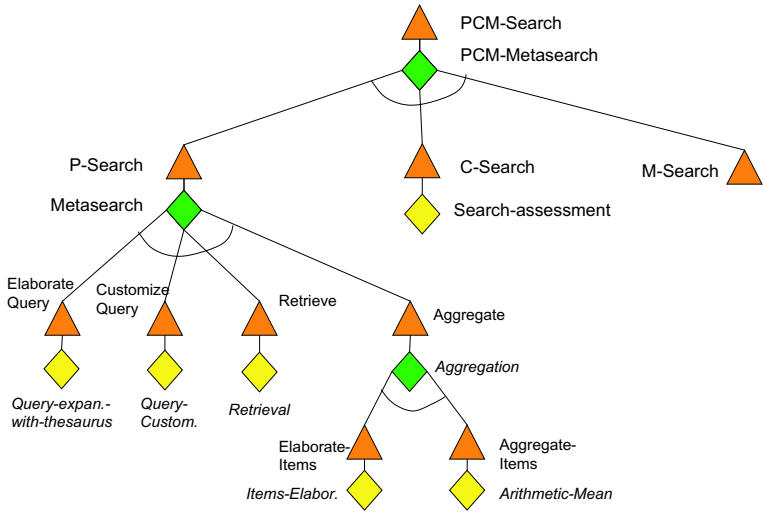

Figure 7.21: Task-configuration with a task in delayed configuration mode

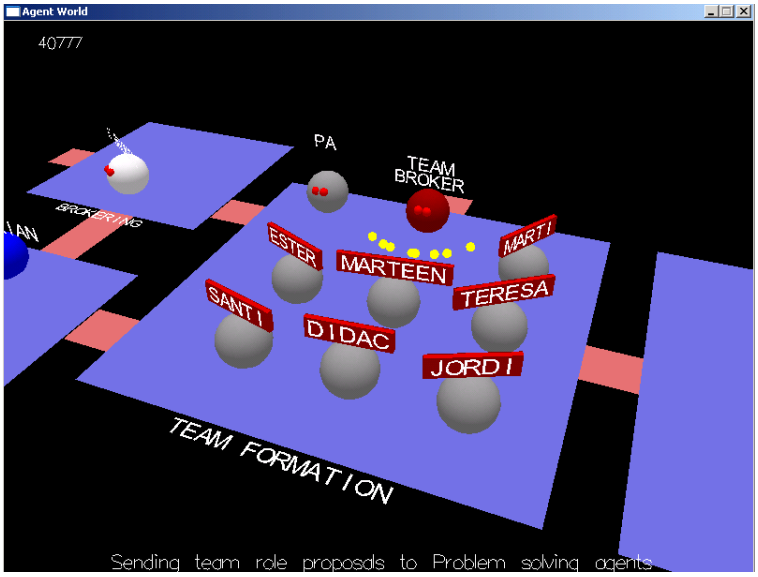

Figure 7.22: Screenshot of a Team Formation scene

whether they accept or refuse such a proposal. PSAs accepting a team-role proposal are considered candidate agents for the corresponding task (recall there is one team-role per task) by the T-Broker.

The T-Broker sends proposals until all the tasks are allocated to some agent or some failure criterion is reached that leads the T-Broker to abort the process without having succeeded. After that, if the task-allocation succeeded, the Team-broker chooses among alternative agents for the same task, and inform selected agents of the result.

Figure 7.23 shows a screenshot of the T-Broker internal state after having obtained candidates for all the tasks. The tasks to be allocated are on the left, while candidate agents are represented on the right. Agents already selected for a task are recognized by a square surrounding them. At the moment shown, the T-Broker has already finished the task-allocation process and is performing the team selection process (selecting agents for each task); for instance, Ester has been just selected for the task Retrieve, to the detriment of Marc, who is also a candidate agent for that task. Notice that the team selection activity proceeds following a bottom-up direction, from the tasks in the leaves of the task-configuration tree upwards, until reaching the top level task. Tasks with delayed configuration, such as M-Search, are not being allocated, since the capabilities to be applied for those tasks are not known yet.

Figure 7.23: Example of the T-Broker internal state

After finishing the team selection process, the T-Broker informs the participating agents on the result of the team selection process, and instructs selected agents about the team-roles they will play. A team-role (§5.3) includes the following information: a task to be carried out, the capability required to solve that task, and all the information required to cooperate with other team mates, which is encompassed by a collection of team-components. The agents obtained as candidates for some task but not selected in the end, are kept in reserve to replace a team-member if it fails during the Teamwork process, just in case.

At the end of the Team Formation scene, the T-Broker sends the resulting *team-configuration* to the PA, which is then ready to initiate the Teamwork scene. Meanwhile, the PSAs selected to participate in the team move to the Teamwork scene (in Figure 7.25 one can see the agent Marc waiting in the Team Formation scene, while all the selected agents have already moved to the Teamwork scene).

7.9.5 Teamwork

The Teamwork scene starts when the PA sends the data of the problem to be solved to the agent assigned the top-level task in the task-configuration, the so called team-leader. In the example, the team-leader is Marteen, who is assigned the task Information-Search, and has to apply the task-decomposer PCM-Metasearch.

The team-leader starts the problem solving activity using the information provided by the team-role for the top-level task, Information Search in our example. The information about which specific capability the team-leader has to apply for solving that task, and the name of the agents to whom delegate a task are described by a *team-role* provided to the team-leader by the T-Broker at the Team Formation scene. Besides this, any agent does the same when requested to play a team-role: consulting the information provided by the R-Broker and applying the specified capability.

Figure 7.24 shows a screenshot of a PSA internal state, indicating the capabilities it is equipped with and the team-roles it has to play, represented by pairs composed of a task and a capability bound to it. If a PSA has to apply a task-decomposer, the information contained in the team-role includes also the subtasks introduced by the task-decomposer and the agents to whom delegate each subtask. Figure 7.24 shows Marti's internal state, which is equipped with three capabilities: Metasearch, Metasearch-without-elaboration, and Modify-Metasearch. Marti has been assigned the team-role associated to the task P-Search, which is bound to Metasearch, a task-decomposer. The figure shows also the names of the agents assigned to each of the subtasks introduced by the task-decomposer Metasearch: Teresa is assigned to Aggregate, Ester to Retrieve, Didac to Customize-query, and Santi to Elaborate-query.

A Problem-Solving Agent (PSA) can form part of different teams at the same time, thus when a PSA receives a request to carry out the task associated to a specific team-role, it has to check whether it is assigned to that team-role and which capability to apply. Next, the PSA checks whether that capability is a skill or task-decomposer: if it is a skill the PSA applies that skill to solve the task and sends the result to the requester; otherwise the capability is a task-decomposer and the PSA will follow the task-decomposer's operational description.

During the Teamwork scene, the control flow follows the performative structure specified by the operational description of the task-decomposers in the task-configuration, as described in §5.6.

Figure 7.25 shows a picture of the teamwork scene, just when the PSA assigned to the retrieve task is sending queries to the Pubmed information source

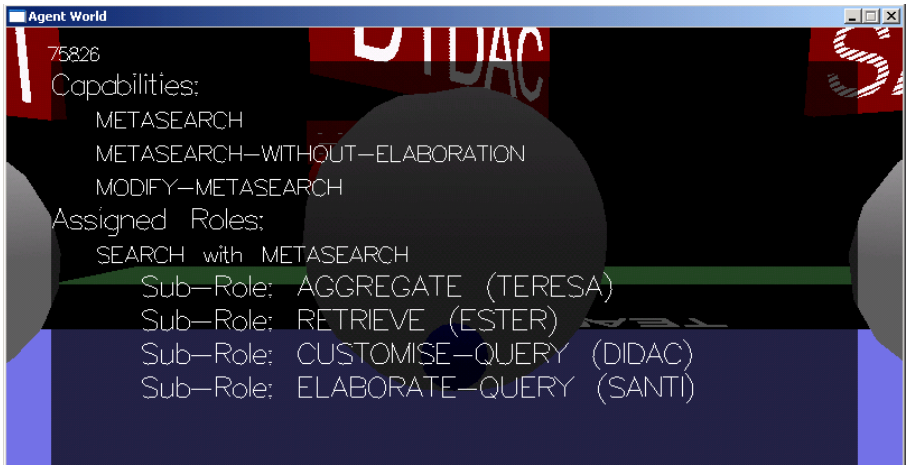

Figure 7.24: Example of a PSA internal state

through the Internet, using the Pubmed wrapper.

If all tasks are solved successfully, the final result is hold by the team-leader, who sends it to the PA, and the Cooperative Problem Solving finishes.

In our example, there is a task that is not bound to any capability because this decision depends on the result of another task, Critique-Search. Consequently, the agent detecting that condition (the agent assigned to the task in top of the delayed task) informs the PA. In our example, that agent is Marteen, responsible for the task Information-Search. Marteen informs the PA of that condition, and provides it with the information required to configure the task with delayed configuration. In the example, the result of P-Search contained a number of items considered insufficient by the capability Search-Assessment. Consequently, Search-Assessment outputs the formula GENERALIZE-QUERY (the reader is referred to §5.7.1 for a more detailed account of this mechanism) to indicate that it is convenient to generalize the scope of the query in order to get more results. Marteen informs the PA that the task M-Search has to be configured using the formula GENERALIZE-QUERY as a new postcondition for the Knowledge Configuration process. Therefore, the PA moves again to the Knowledge Configuration scene to request the K-Broker to configure the task M-Search using the new postcondition.

Figure 7.26 shows a screenshot of the K-Broker internal state during the new Knowledge Configuration process. Notice that the K-Broker has selected the capability Query-generalization to solve the task Adapt-query, since it is the only capability specifying the formula GENERALIZE-QUERY as a postcondition.

After configuring the task Modify-Search, the PA engages a Team Formation scene again to form a team for that task. The agents involved in the original team are waiting in the Teamwork scene, that remains active, but in the meantime, they are requested again to form part of the new team responsible for

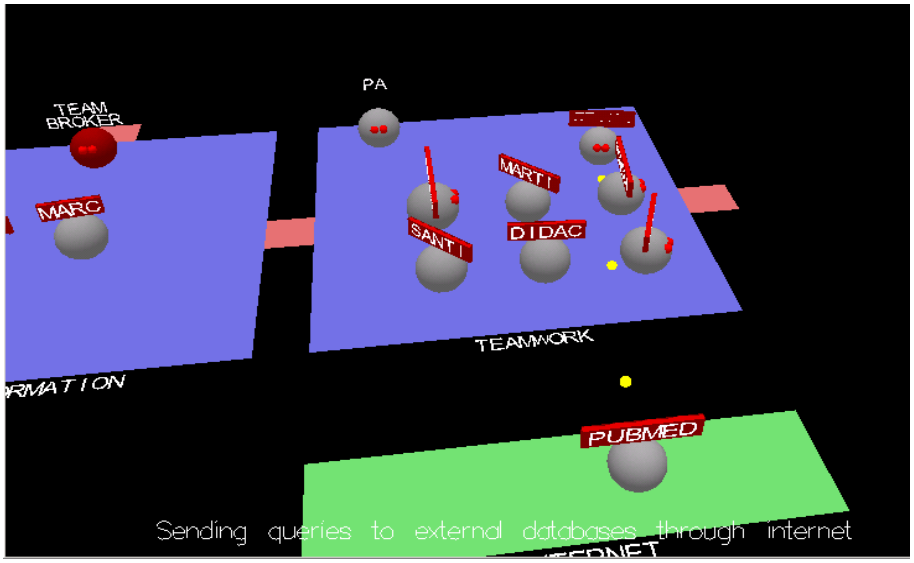

Figure 7.25: Screenshot of the Teamwork scene

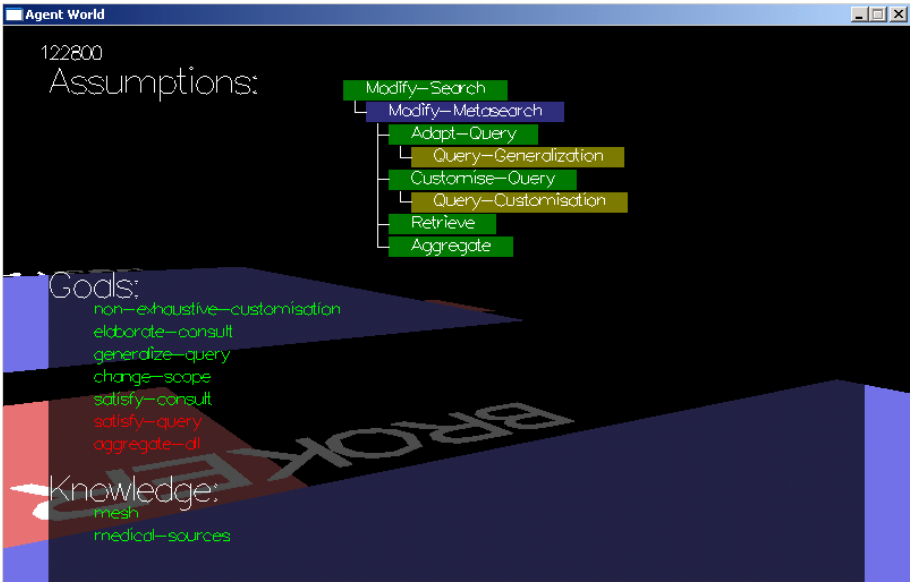

Figure 7.26: Example of the K-Broker internal state for a delayed task

the Modify-Search task. Recall that an agent can participate simultaneously in several scenes, though the Agent World monitoring tool is unable to represent that condition.

Finally, the PA sends the result back to the GUI (or to the agent that originated the CPS process). Figure 7.27 shows the results for the consultation example described above (§7.9.2).

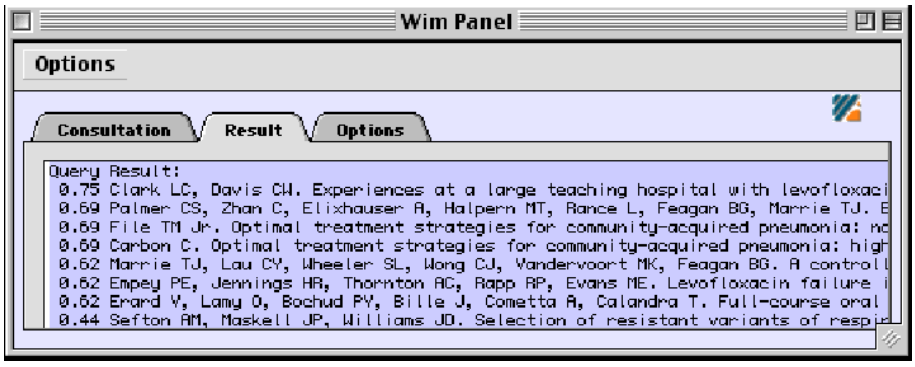

Figure 7.27: Team broker internal state

If the results have to be sent to an external agent using FIPA-ACL, the PA encodes them using XML and sends them within the content of a FIPA inform message to that agent.

```
(inform
    :sender     (agent-identifier :name WIM-PA@\wim\.iiia.csic.es:7778/NOOS)
    :receiver   (set (agent-identifier :name uva-agent@a1136.fmg-uva-nl:1099/JADE))
    :reply-with     cps-request18236
    :encoding   String
    :language   FIPA-SL0
    :ontology   WIM-Ontology
    :protocol   FIPA-request
    :conversation-id    cps18236
    :content
       (result
          (action \ldots )
          (Problem-Solution:
             :encoding  <xml? version="1.0" encoding ="ISO-8859-1"?>
         :value
            <scored-items>
               scored-item>
                 <IDENTIFIER>PMID10683053</IDENTIFIER>
                        <RANKING> 0.56</RANKING>
                        <TITLE>A controlled trial of a critical pathway for treatment of
                    community-acquired pneumonia.</TITLE>
                        <AUTHORS>Marrie TJ, Lau CY, Wheeler SL, Wong CJ, Vandervoort MK,
                      Feagan BG.</AUTHORS>
               </scored-item>
               <scored-item>
                 <IDENTIFIER>PMID10418757</IDENTIFIER>
                        <RANKING> 0.34</RANKING>
                        <TITLE>Selection of resistant variants of respiratory pathogens
                    by quinolones.</TITLE>
                        <AUTHORS>Sefton AM, Maskell JP, Williams JD.</AUTHORS>
               </scored-item>
```

```
</scored-items>))))
```

7.10 Other experiments

The WIM application has been developed in the framework of the European Project IBROW, which stands for Intelligent Brokering Service for Knowledge-Component Reuse on the World-Wide Web. The objective of IBROW is to develop intelligent brokers that are able to distributively configure reusable components into knowledge systems through the World-Wide Web, which leads interdisciplinary research on areas such as heterogeneous DB, interoperability and Web technology with knowledge-system technology and ontologies. Concerning the feasibility and the applicability of the IBROW ideas, the project has encouraged the development of interlibrary applications involving components implemented by different partners. Some of these efforts have focused on agent technology, whilst other efforts have turned to new technological developments such as the Semantic Web Services.

Now we are going to review two applications: (1) an application concerning two libraries implemented by agents running on different platforms, and (2) a Semantic Web Service communicating with the WIM application.

7.10.1 Inter-library application

The idea of inter-library applications is to configure applications that involve problem solving components from different libraries, and running in different platforms as well. In such an scenario, three types of interoperation problems arise: semantic, syntactic and relative to the interaction protocols.

At the semantic level, in order to compare (match) specifications from different libraries, either a shared ontology specifying the concepts used by the different libraries or a collection of mappings between the different ontologies is required.

At the syntactic level, a common language, a encoding data format and a serialization mechanism to send and receive data between components from different libraries are necessary, specially when two libraries are implemented over heterogenous platforms. For instance, in the IBROW project we have connected the ORCAS agent platform with a JADE platform through the FIPA-Mediator, using the FIPA-ACL, and either XML or RDF to encode the content of a message.

Finally, agents must share some interaction protocol in order to communicate meaningfully. Therefore, we have developed interaction protocols based in the FIPA Request-Inform protocol and using the FIPA Agent Communication Language.

The interlibrary application involves two libraries: one library is the one provided by the WIM application, running over the NOOS agent platform, which uses LISP; the other library is intended for document classification tasks, has

been developed by the SWI group, from the UVA University, and is implemented over the JADE platform, which uses Java. The are two agents interconnecting both, the FIPA-Mediator running in the NOOS platform, and the NOOS-Proxy running in the SWI agent platform. These (pseudo) agents implement the communication and the data translation between both agent platforms using FIPA-ACL as the communication language, and SL0 as the content language. Figure 7.28 shows the interconnection of both platforms and the agents responsible for the interoperation: a Broker and a User Agent in the SWI platform, and a Librarian and a Personal Assistant in WIM.

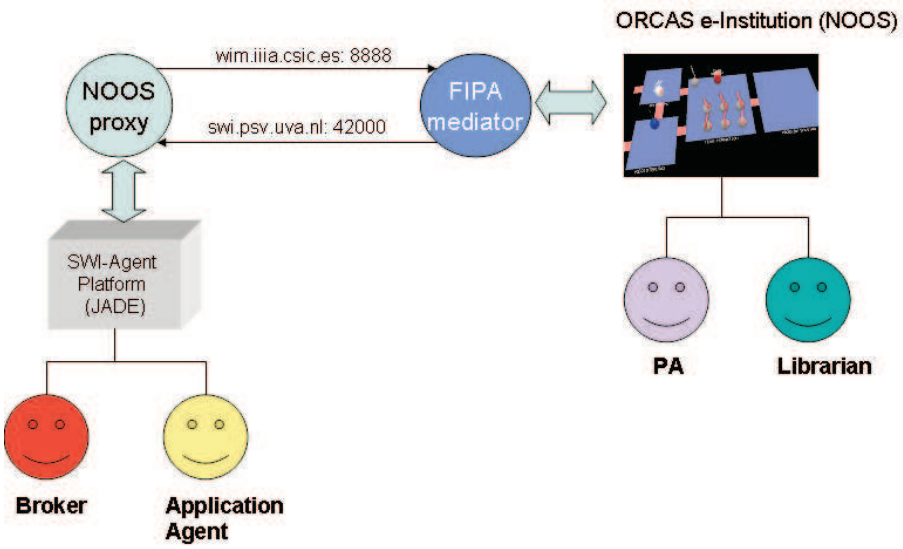

Figure 7.28: Interlibrary application

The SWI Broker asks the WIM Librarian about the tasks and capabilities available in WIM to check whether a configuration of the application task using WIM components is possible. The User Agent interacts with the user to get a specification of the problem and interacts with the SWI Broker to obtain a configuration of components for the application task, that is called Search & Classify. The Search & Classify task is decomposed into three subtasks by the Search, Retrieve & Classify task-decomposer, as shown in Figure 7.29.

The WIM PA acts as a mediator between the SWI broker and the ORCAS brokers: The Knowledge-Broker and the Team-Broker. The idea is that the WIM brokers operate locally over the tasks to be performed by WIM agents, while the global task is configured by the SWI broker, that delegates the configuration and execution of some tasks to WIM agents.

The ORCAS e-Institution in which the WIM application is deployed offers some services to external agents, including the following (Appendix F):

1. *Brokering*: Obtaining a task-configuration according to a specification of

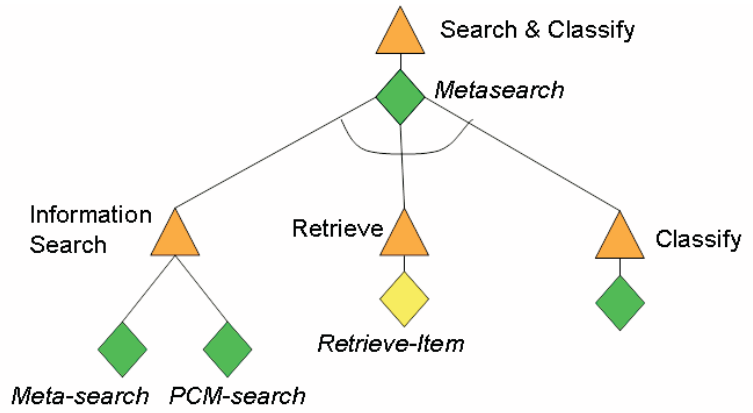

Figure 7.29: Interlibrary application

problem requirements.

2. *Team formation*: forming and instructing a team of agents with the capabilities required by a task-configuration

3. *Teamwork*: performing the teamwork activity to solve a problem instance, given a team-identifier of a previously instructed team.

4. *Cooperative Problem-solving*: this service comprises all the previous services within a single request-inform protocol.

5. *Retrieve specification*: this service provides a pattern-based retrieval service to gather component specifications from the library.

Figure 7.30 shows the subtasks and data involved by the interlibrary application task, Search & Classify. The goal of this task is to search bibliographical references, and then retrieve all extra information available for some references in order to classify them. This task is decomposed into tree subtasks by our task-decomposer: Information-search, Retrieve-extra-info and Classify.

The input data to the Information-search task is a query, and the result is set of items containing bibliographical references, and ranked according to the relevance to the query. These items include only the basic information: an identifier, a title and the list of authors. Next, some of these items are used as inputs of the Retrieve-extra-info task, which is able to retrieve all the information available for a reference, like the language, subject (keywords), abstract, year and so on. Since several items may be selected for classification, then several instances of this task may be required, one for each item selected. Finally, the retrieved items are clustered and classified by some classification method in the SWI library, using the information provided by the WIM task Retrieve-extra-info.

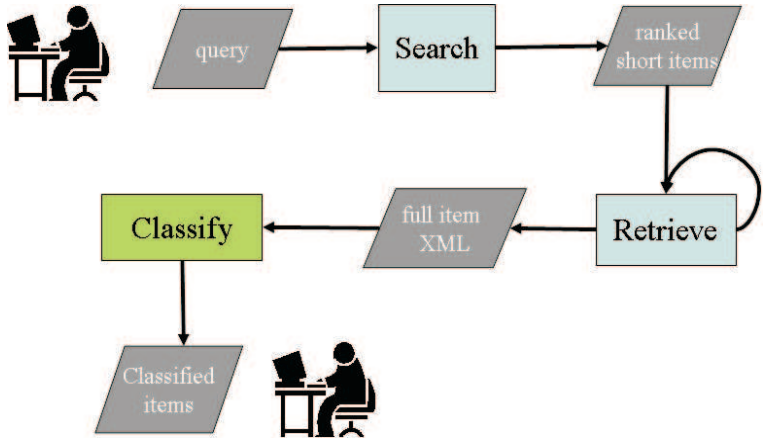

Figure 7.30: Interlibrary application

7.11 Conclusions

Finding relevant information can be described only with respect to the purpose for which this information is sought, for the kind of use we have in mind for that information. However, classical information retrieval (IR) based on keywords assumes all the information provided by the user is flat and in the form of keywords. The documents being searched are abstracted as vectors of keywords to be compared with the "document query" of the user also abstracted as a vector of keywords. This approach focuses on the notions of *content* and *relevance*: the content of documents are expressed as vectors of keywords and the similarity measure that compares them with the query vector expresses the degree of relevance of each document to that query.

Let us consider the query example we have used in this article: a user is interested in information about *guidelines* on the use of *levofloxacin* in the treatment of *pneumonia*. The keyword based approach would recommend simply to use the keywords `(guidelines, levofloxacin, pneumonia)` as a flat representation of the user request. However, we know that the user is in fact talking about a substance (levofloxacin), a disease (pneumonia), and a type of information searched (guidelines).

The approach taken in the Web Information Mediator has been to provide a bridge (a mediation service) between the user's expression of the kind of information to be found and the existing repositories of medical information that use keyword-based retrieval. WIM distinguishes between keywords that define content (e.g. searching documents *about* pneumonia) from categories that define the intended *use* of the information (e.g. using information on *guidelines* instead of, say, *diagnosis*). Categories are thus used to transform the initial user request into a collection of keyword-based queries that include the knowledge needed to find occurrences of keywords related to that category. As we have

seen, different keywords related to *guidelines* are used in these queries—while a keyword-based retrieval would have only used the keyword `guidelines`. The retrieved documents are ranked in accordance to how useful they are for the intended purpose of the category *guidelines*, but they also include documents indexed as *clinical protocols*. Other utility criteria used in WIM are related to specifying a broad and costly search towards a narrow and fast search. We have shown no example here but the idea is very similar in that a thesaurus is used (instead of a category) to elaborate further queries using synonyms and related words. The criterion of utility is here related to the type of search and can be combined with the criteria based on EBM.