jCOLIBRI 2

Case-Based Reasoning Framework

ALADIN Contribution Guide

Abstract LAyer for DIstributed CBR INfrastructures

September 23, 2009

Document version 1.0

GROUP FOR ARTIFICIAL INTELLIGENCE APPLICATIONS
UNIVERSIDAD COMPLUTENSE DE MADRID

1 What is this contribution for?

This contribution aims to provide a reference framework for developing distributed CBR applications with jCOLIBRI. ALADIN contains the main interfaces required to build (almost) any distributed CBR system. It provides an *abstraction layer* that must be extended depending on the features of the system: communication with sockets, JADE, etc. [1]

This package contains the interfaces that must be followed in future implementations of distributed systems in jCOLIBRI. These interfaces try to complain the IEEE FIPA[2] standard for distributed systems. This way, we guarantee a common root that will ease the integration of different techniques.

The FIPA standard defines an abstract architecture for multiagent systems, that is independent of the implementation language, promoting interoperability and reusability. In our contribution, we have used a reduced version of this architecture, trying to exploit the advantages of the standard.

This abstract layer defines an architecture where each agent is a CBR system. These agents will communicate to send/receive queries, cases, solutions, etc.

2 Entities of the contribution

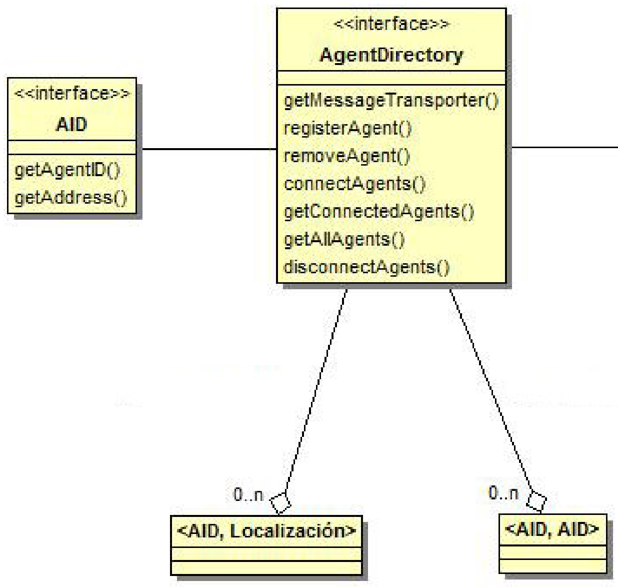
The ALADIN abstract layer for building distributed CBR systems is composed of 6 main entities, described next.

2.1 Agent Directory

The role of this entity is to provide a directory or registry where agents can register to be localized by other agents. When an agent of the distributed system wants to communicate with other agent it must query the `AgentDirectory`. The `AgentDirectory` keeps a table with tuples `<AID,localization>` with every agent in the system. AID (`AgentIdentifier`, see Section 2.3) is an unique number that identifies an agent in the distributed system. The `localization` entry will vary depending on the implementation. Moreover, the `AgentDirectory` may keep information about the topology of the system's network of agents. This list is another table with `<AID, AID>` tuples representing the connections between agents.

[1]This abstraction layer has been defined in the Master Project of Sergio Gonzalez Sanz, who performed an exhaustive revision of current literature on distributed CBR systems. This document is available only in Spanish, but copies are available by contacting with the jCOLIBRIteam.

[2]http://www.fipa.org/, standard SC00001L



The `AgentDirectory` has its own AID because it acts as another agent in the system. Additionally, it will use a `MessageTransporter` (see Section 2.6 component to answer the queries received from other agents.

The operations that an `AgentDirectory` must support are:

- Register an agent in the system.
- Remove an agent from the system.
- Keep the connections between agents in the systems.
- Remove connections between agents.
- List agent's connections in the system.

Figure 2.1 shows an UML diagram with the relationships of this entity.

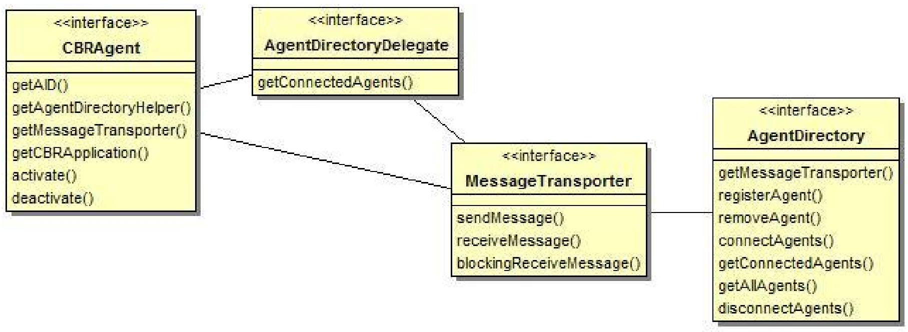
2.2 AgentDirectoryDelegate

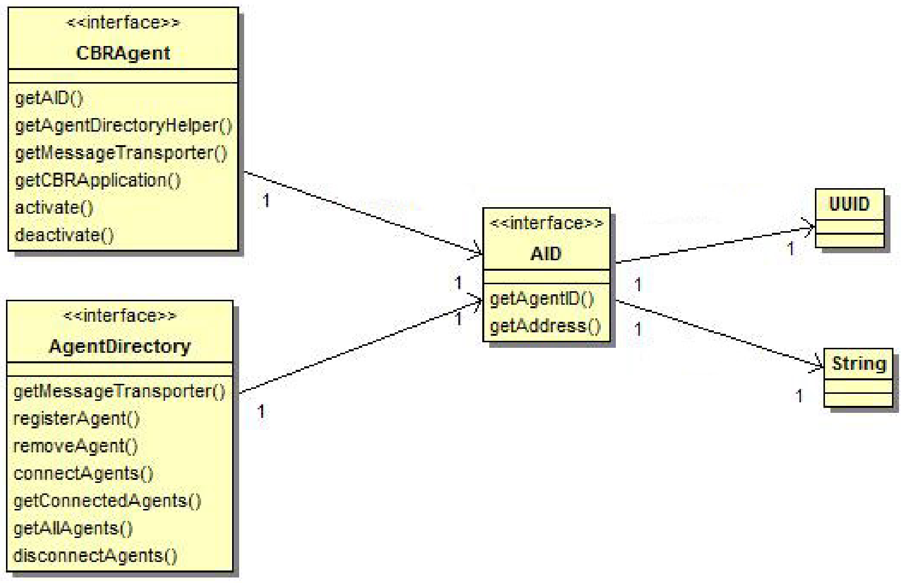
The role of the `AgentDirectoryDelegate` is to provide access to the `AgentDirectory`. This is, enables an agent to know which other agents is connected to. The `AgentDirectoryDelegate` entity is used by every `CBRAgent` (see 2.4) in the system to connect with the `AgentDirectory`. Therefore, every `CBRAgent` will have an `AgentDirectoryDelegate`.

The operation provided by the `AgentDirectoryDelegate` is:

- Contact the `AgentDirectory` and return which agents are connected to a concrete agent (who owns the delegate).

Figure 2.2 shows an UML diagram with the relationships of this entity.





2.3 AID

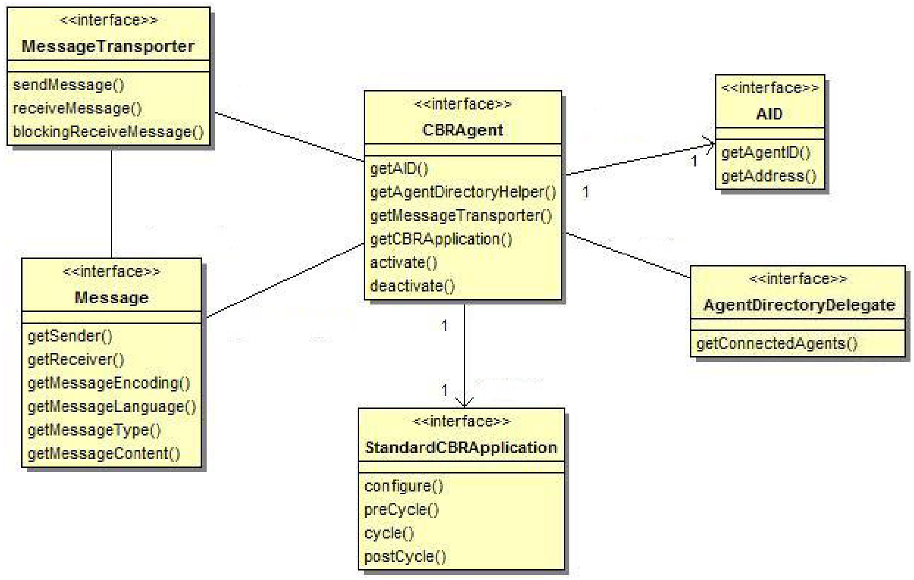
The `AID` (or `AgentID`) entity identifies unequivocally every agent in the distributed system. The `AID` assigned to an agent is not modified during the life cycle of that agent. This entity is used in the system to keep track of the topology of the system.

Figure 2.3 shows an UML diagram with the relationships of this entity.

2.4 CBRAgent

An agent is a computational process that is autonomous and is able to communicate with other agents. In the case of `CBRAgents`, they contain independent CBR systems that implement the jCOLIBRIinterface (`StandardCBRApplication`). Every `CBRAgent` has an unique AID to be identified, and an `AgentDirectoryDelegate` to query the `AgentDirectory` for the agents thatis connected to. A `CBRAgent` also uses a `MessageTransporter` entity to send and receive messages to/from other agents.

The basic operations that a `CBRAgent` must support are:



- Provide the contained CBR system access to every associated entity: AID, `AgentDirectoryDelegate` and `MessageTransporter`.
- Enable and disable the agent in the distributed system.

Figure 2.4 shows an UML diagram with the relationships of this entity.

2.5 Message

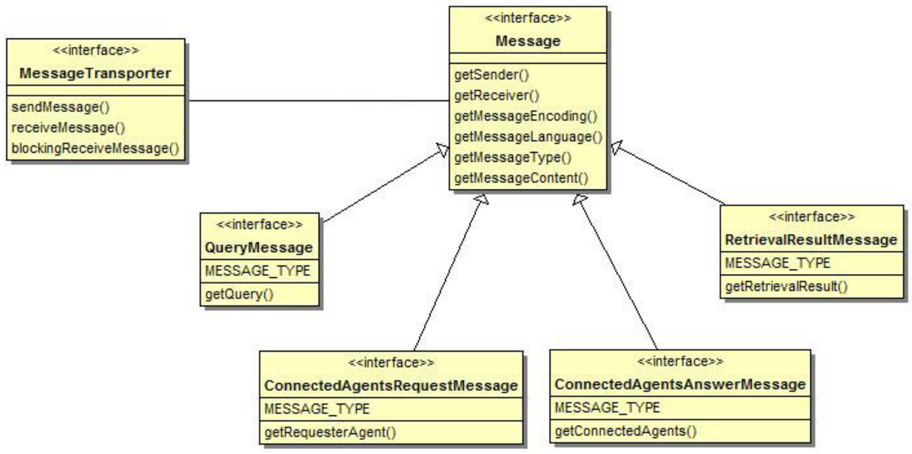
A Message is an individual communication unit between two or more agents. A Message includes information about the kind of communication, the `AID`s of the sender and receivers, and the data being transmitted. Depending on the distributed system, there will be different implementations of `Message`.

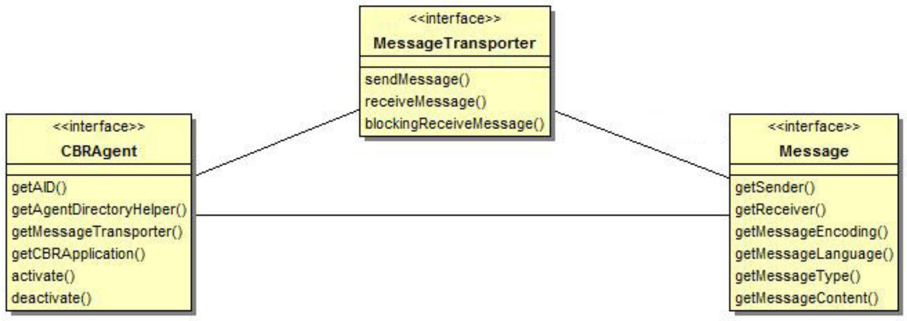
This entity is used by a `CBRAgent` to communicate with other `CBRAgent`, and is the entity handled by the `MessageTransporter` to send and receive information. It must support the following operations:

- Identify the sender and receivers through the `AID`s.
- Identify the language/codification of the contained data.
- Provide access to the contained data.
- Provide a description of the kind of message.

The jCOLIBRI's ALADIN contribution provides some standard messages that are commonly found in most distributed systems:

- `ConnectedAgentsRequestMessage`: Message to request connected agents.





- `ConnectedAgentsAnswerMessage`: Message with the agents that are connected to a given one.
- `QueryMessage`: Message that encapsulates a query for the CBR system.
- `RetrievalResultMessage`: Message that encapsulates a collection of `RetreivalResult` objects (containing the retrieved cases).

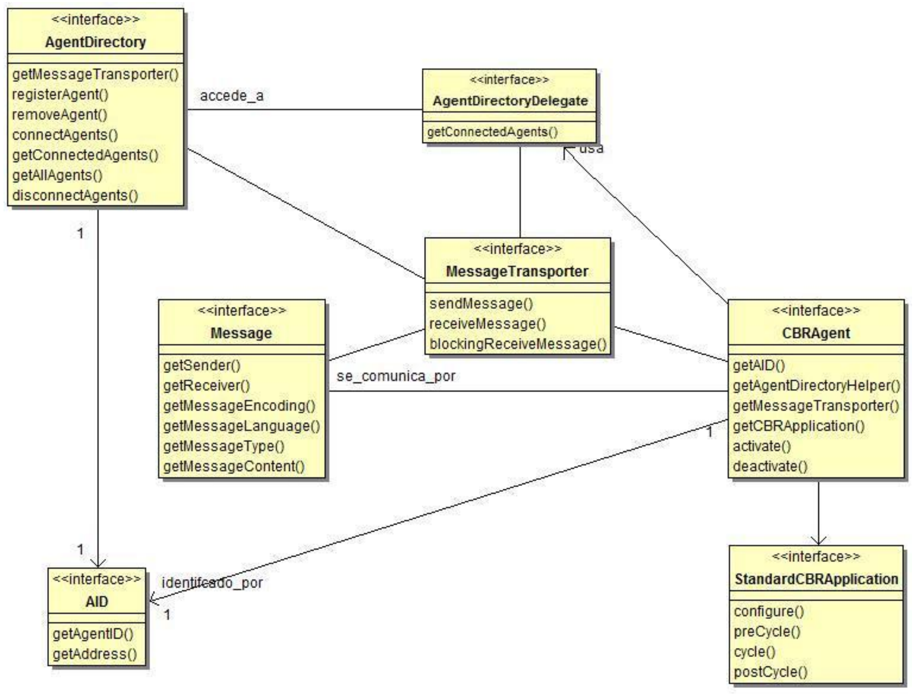
Figure 2.5 shows an UML diagram with the relationships of this entity.

2.6 MessageTransporter

The `MessageTransporter` entity provides a communication service in the multi-agent system. This entity supports `CBRAgents` to send and receive messages. The implementation of this entity is very depending on the concrete distributed application being developed, because there are several possibilities: sockets TCP/IP, shared memory, etc. These entities must provide (at least) the following operations:

- Send a message.
- Receive a message.

Figure 2.6 shows an UML diagram with the relationships of this entity.



2.7 Global overview

To complete the description of the ALADIN contribution we include a general UML diagram that offers a global view of the relationships between the entities described previously.

Figure 2.7 shows this general overview of the entities. The UML diagram illustrates the central role of the `CBRAgent` entity, which contains a `StandardCBRApplication`. Every `CBRAgent` communicates through Messages that are sent and received with the `MessageTransporter` entity. `CBRAgents` are described by an AID object that is used to localize and describe the topology of the distributed system.

Finally, the `AgentDirectoryDelegate` eases the access to the `AgentDirectory` from a `CBRAgent`. `AgentDirectoryDelegate` objects will use the `MessageTransporter` to communicate with the `AgentDirectory`.