Adding an Ontology to a Standardized QoS-Based MAS Middleware

José L. Poza, Juan L. Posadas, and José E. Simó

Institute of Industrial Control Systems
Polytechnic University of Valencia
Camino de Vera s/n, 46022, Valencia, Spain
jopolu@ai2.upv.es, jposadas@ai2.upv.es, jsimo@ai2.upv.es

Abstract. In a Multi-Agent system, middleware is one of the components used to isolate control and communications. The use of standards in the implementation of an intelligent distributed system is always advantageous. This paper presents a middleware that provides support to a multi-agent system. Middleware is based on the standard Data Distribution Services (DDS), proposed by Object Management Group (OGM). Middleware organizes information by tree based ontology and provides a set of quality of service policies that agents can use to increase efficiency. DDS provides a set of quality of service policy. Joining quality of service policy and the ontology allows getting many advantages, among others the possibility of to conceal some details of the communications system to agents, the correct location of the agents in the distributed system, or the monitoring agents in terms of quality of service. For modeling the middleware architecture it has used UML class diagrams. As an example it has presented the implementation of a mobile robot navigation system through agents that model behaviors.

1 Introduction

One of the biggest problems in the distributed systems is the efficient location of information. Most times, the view that the agents have of the system is rather strict, and depends entirely on communications system. Abstract details of the system to the agents, provides greater flexibility, adaptability and scalability of the system. Also, one of the most significant technological challenges is the management of peer-to-peer quality of service (QoS) for component-based distributed intelligent control systems.

These aspects of the distributed systems, go beyond the real time requirement, and involve considerations such as: availability of computational resources, security, co-operative control algorithms, stability, task control performance and management of redundant information. Nowadays, the design of communication systems does not offer an abstract view of the system and a complex QoS, just very simple features of QoS like message sequencing, traffic congestion relieving, and so on. The union of ontology and quality of services policies provides by the middleware, offers to agents a meta-information attractive to optimize their processes.

The rest of the paper has been organized as follows: Second section presents essential concepts about middleware, quality of services and ontology. Third section explains the standard of communications DDS proposed by OMG. Next section describes the architecture modelled in UML. This model unifies concepts of message queues, quality of service policies and the ontology. Fifth section shows an example of the use of ontology in robot navigation architecture. Finally presents concluding remarks and future of the project.

2 Middleware, Quality of Service and Ontology

Most of the communications systems that provide support to the distributed control architectures need a module that hides some details of the communications components. Usually, when this module is separated from control components, is known as "middleware". To provide to control components, the services needed to increase efficiency of communication is the main responsibility of middleware. Among the required services are: identification of components, authentication, authorization, hierarchical structuring or components mobility.

Above all, technology underlying programming like objected-oriented programming, component-based programming or service-based programming, partly determine control architecture and its ability to provides more QoS [1]. There are a lot of interfaces and tools for developing a middleware. Some of the tools like JMS [2] and MSMQ [3] are generic protocols, and widely used on distributed systems.

In distributed multi-agent systems some components need to be adapted to the communication interfaces For example, if communications are based on CORBA [4], the multi-agent system must be implemented with the object-oriented programming technology. To avoid the use of a particular technology is common to use standardized protocols like FIPA [5].

QoS defines a set of parameters for evaluation of a service offered. In the field of control architectures there are many definitions of quality of service. From the viewpoint of processing, QoS represents quantitative and qualitative characteristics of a distributed system. These characteristics are needed to achieve the functionality required by an application.

From the viewpoint of communications, QoS is defined as all the features that a network has to meet for message flow [6]. The term ontology has its origin in philosophy, and has been applied in computer science research [7]. The core meaning within computer science is a model for describing the world that consists of a set of types, properties, and relationship types [8].

3 Data Distribution Service

Data Distribution Service (DDS) provides a platform independent model that is aimed to real-time distributed systems. DDS is based on publish-subscribe communications paradigm. Publish-subscribe components connect information producers (publishers)

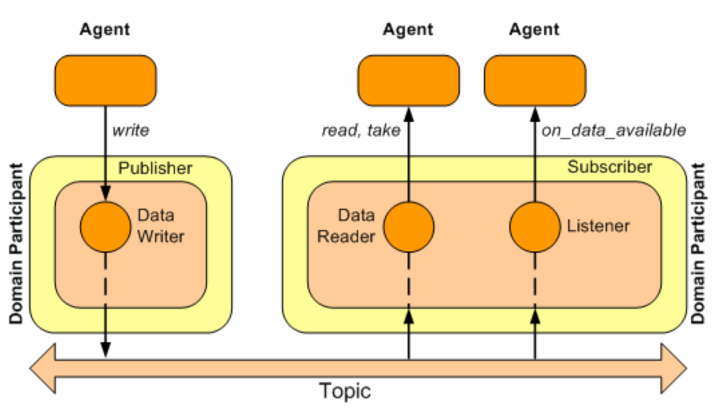

Fig. 1. Overview DCPS components from the DDS model

and consumers (subscribers) and isolate publishers and subscribers in time, space and message flow [9]. To configure the communications, DDS uses QoS policies. A QoS policy describes the services behavior according to a set of parameters defined by the system features or by the administrator. Consequently, service-oriented architectures are recommended to implement QoS in its communications modules.

DDS specifies two areas: Data-Centric Publish-Subscribe (DCPS) which is responsible for data distribution and DLRL which is responsible for adjusting the data to local level of applications. DLRL area is optional due to the DCPS components can work directly with the control objects without data translations. DCPS has a large number of component and some of them are required in any implementation. This is presented in figure 1.

When a producer (component, agent or application) wants to publish some information, should write it in a "Topic" by means of a component called "DataWriter" which is managed by another component called "Publisher". Both components, DataWriter and Publisher, are included in another component called "DomainParticipant". On the other hand, a Topic cans delivery messages to both components: "DataReaders" and "Listeners" by means of a "Subscriber". When the application requires it, DataReader provides the messages instead of a "Listened". Messages are sent without waiting for the application requires.

4 Formal Model

Among formal specifications, Unified Modelling Langage (UML) is the language of modelling and formal software systems descriptions best-known [10].UML is supported by the Object Management Group (OMG). Consequently, is appropriate use UML to describe the Middleware internal architecture.

Figure 2 shows a formal description of the middleware architecture by means of a UML class diagram. "Entity" is class base for all components, except for the QoS policy. Each component can have associated several QoS policies.

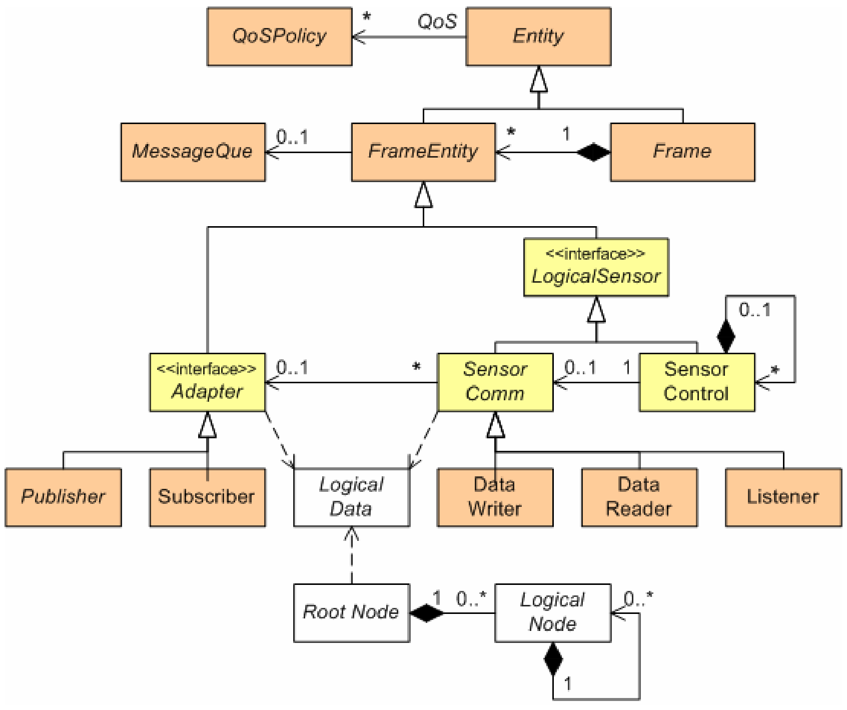

Fig. 2. UML class diagram of the middleware with the ontology support

The role of a "LogicalData" is the same that "Topic" in DCPS. When a "Logical Sensor" does not have an associated "Adapter", then is a control component, and can be associated with others control components.

The ontology is implemented from the abstract class "LogicalData". This class provides the logical datas to agents. Through a logical data, agents have access to information. The root node contains the sequence of logical nodes that make up the

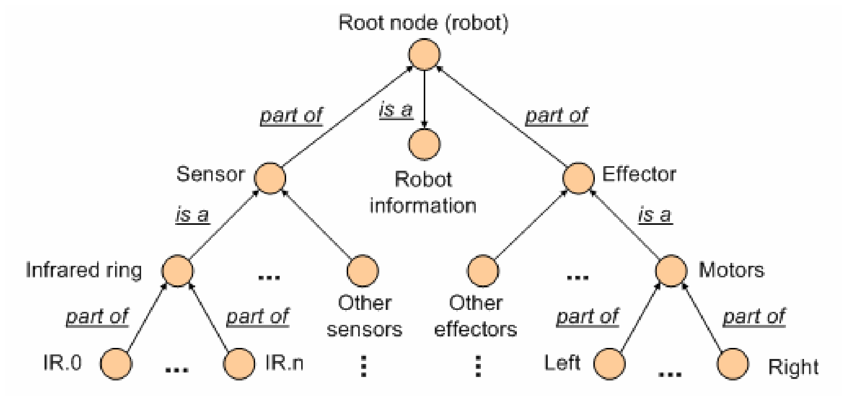

Fig. 3. Example of mobile robot system ontology

ontology and each logical node has a property that relates it to other. Initially only have been defined relations "is a" and "part of", through these relations, an agent can learn the system structure and act accordingly to their interests.

The use of ontology, as a method of information access, is useful to agents because it provides two important functions. The first of these functions is the system accessing interface, either to receive data from the sensors, to send control actions. The second of these functions, is to get a representation of the system that allows to agents to learn. An agent can learn about the information to communicate with other agents and the system structure that provides such information.

The structure of the system is interesting because agents can ask to the communications system about questions like “what kind of sensors are installed on the robot”. In addition, an agent can be connected to a specific data set like “warn only when proximity sensors above a certain value”. Moreover, the structure allows an agent to write to the data belonging only to a specific category like “stop all the wheels”.

Joining ontology with the quality of service policies provides other benefits. An agent can search process nodes based on both criteria. For example, is possible to search a sensor that provides data with a deadline less than a specific value or a motor driver with a message queue with of a specific buffer size. Based on the previous model, simple robot navigation architecture has been developed. The architecture has two distinct parts: control and communications. Quality of service joints both levels [11]. Communications layer manages the ontology and offers its services through the DDS interface [12].

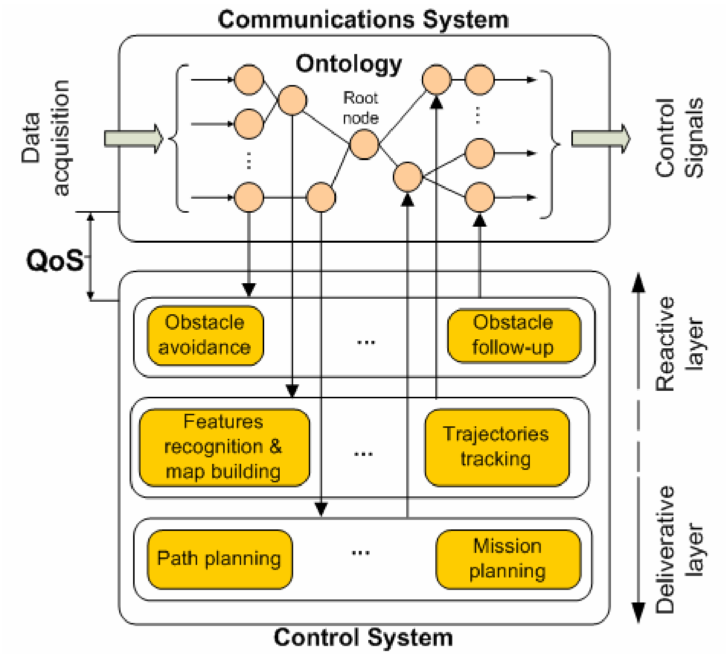

Fig. 4. Robot navigation architecture implemented with the FSA-Ctrl architecture

5 Case of Use: Mobile Robot Architecture

Usually, robot navigation architectures are organized in two layers: deliberative and adaptive or three layers with an intermediate layer. No such differentiation in the FSA-Ctrl architecture due to agents can be auto-organized. Logical data of the ontology differentiates deliberative agents from reactive. Usually deliberative agents are connected to logical data near the root node, and reactive agents are connected to logical data away from the root.

Figure 5, shows an example of ontology used to describe the distributed system of sensors for a mobile robot and in figure 4, shows an example of the use of basics behaviours of navigation architecture. One of them, like the obstacle avoidance or obstacle tracking, can be considered as reactive, since the decision doesn't imply the query to a pre-established plan, and has high temporal restrictions of data.

Other behaviours, such as route path planning, may be considered as deliberative because they have more time limits. When determining behaviour in the robot navigation system, they are associated with logical data. The depth in the ontology of an

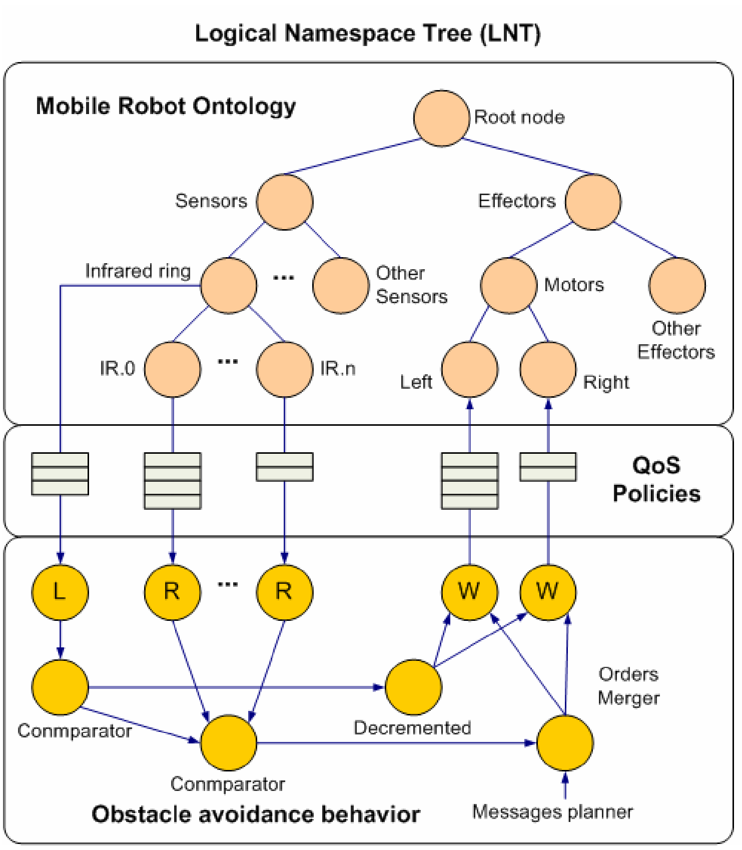

Fig. 5. Example of ontology to provide information to a obstacle avoidance behaviour

agent connection to a logical data of the middleware provides information about if the agent prefers the reactive or the deliberative layer.

This organization may change depending on the system needs. Sensors have been organized according to the type. Obstacle avoidance agent uses the infrared ring. In this case, actuators are the motors of the robot and agents can write the desired speed. Through the writing on each logical data logical, left or right motor, agent can provide a turn in either direction.

A reactive agent writes data to motors to avoid an obstacle and a deliberative agent writes data to maintain a previously planned path. Quality of service policies differentiates the priority of the reactive agent in front of the deliberative agent.

The "n" infrared sensors that make up the infrared ring are grouped into a logical node called “infrared ring”, as an infrared sensor value exceeds a threshold the logical data is activated, and notifies this change to a “listener”. If some obstacle avoidance agent is connected to this logical data, automatically decreases the speed, without know what specific sensor has sent the alarm.

Moreover, a small-distance path planner agent receives the same message, but this agent request the specific distance to every infrared sensor and calculates the new path to avoid the obstacle. The frequency that messages are sent to agents is not the same to the “Listener” that to de “DataReaders”, the quality of service defined by the designer will determine this aspects.

6 Conclusions

This article has presented the internal architecture of a middleware with QoS support and ontology to organize the information, in order to facilitate the work of agents. Figure 6, shows an application in Visual C that has been developed to design the ontology and create the specified service to the robot. Currently, system is in stage of simulation to determine what set of quality of services parameters are more appropriate to optimize the performance of a home automation system. Results will be presented in future publications.

The architecture is based on the DDS standard model proposed by OMG. Use QoS policies provided by the DDS model, and ontology to hide system details, allowing the system to increase its performance. The middleware can be used to implement various systems. Agents can be reactive or deliberative, only the logical data connections, determine the layer in which the agent works. The hierarchy provided by the ontology, in addition to the quality of service can be used to self-organize agents by means the middleware.

The advantages of the system lie in the possibility to organize information hierarchically by means the ontology. Quality of service provides a mechanism for agents, that allows a self-organized distributed system. Weakness lies in the loss of efficiency typical of a middleware. The use of the standardized DDS interface to communicate agents can be considered a disadvantage if the multi-agent system uses another communication standards, like CORBA or FIPA, but the use of a standard it is always desirable.

Acknowledgements. The MAS architecture described in this article is a part of the coordinated project SIDIRELI: Distributed Systems with Limited Resources. Control Kernel and Coordination. Education and Science Department, Spanish Government. CICYT: MICINN: DPI2008-06737-C02-01/02.

References

1. Coulouris, G., Dollimore, J., Kindberg, T.: Distributed systems, concepts and design, 3rd edn. Addison Wesley, Reading (2001)
2. Hapner, M., Sharma, R., Fialli, J., Stout, K.: JMS specification, vol. 1.1. Sun Microsystems Inc., Santa Clara (2002)
3. Lewis, R.: Advanced Messaging Applications with MSMQ and MQ Series. Que Publishing (1999)
4. OMG. Real-Time Corba Specification version 1.1. Document formal /02-08-02 (2002)
5. FIPA. Specfication. Part 2, Agent Communication Language. Foundation for Intelligent Physical Agents (1997)
6. Vogel, A., Kerherve, B., von Bochmann, G., Gecsei, J.: Distributed Multimedia and QoS: A Survey. IEEE Multimedia 2(2), 10–19 (1995)
7. Smith, B.: Beyond concepts, or: Ontology as reality representation. In: Formal Ontology in Information Systems (FOIS 2004), pp. 73–84 (2004)
8. Gruber, T.R.: Toward Principles for the Design of Ontologies Used for Knowledge Sharing. International Journal Human-Computer Studies 43(5-6), 907–928 (1995)
9. Pardo-Castellote, G.: OMG Data-Distribution Service: architectural overview. In: Proceedings of 23rd International Conference on Distributed Computing Systems Workshops, Providence, USA, vols. 19-22, pp. 200–206 (2003)
10. Object Management Group (OMG). Unified Modeling Language Specification, v1.4.2, ISO/IEC 19501 (2001)
11. Poza, J.L., Posadas, J.I., Simó, J.E.: Distributed agent specification to an Intelligent Control Architecture. In: 6th International Workshop on Practical Applications of Agents and Multiagent Systems, Salamanca (2007)
12. Poza, J.L., Posadas, J.l., Simó, J.E.: QoS-based middleware archi-tecture for distributed control systems. In: International Symposium on Distributed Computing and Artificial Intelligence, Salamanca (2008)