

FEMAS: An Ontology-based Broker Architecture for e-commerce Multi-Agent Systems.

Hugo Perez Santangelo

Facultad de Tecnología Informática
Universidad Abierta Interamericana
Buenos Aires, Argentina
hugo.perez@vaneduc.edu.ar

Abstract. The agents metaphor, expressed theoretically in the Distributed Artificial Intelligence field of Multi-Agent Systems, has been awakening the interest of the commercial software companies, because new emergent technologies like e-commerce follow the same metaphor from the consumer's perspective. Researchers in that field has been developing useful models that can be used to construct commercial servers, but the computational complexity problem and the lack of general models and architectures, sometimes make the technological transfer between researchers and commercial companies impracticable. State of the art is to put the focus on ontologies in order to create more general business models, but there are some software engineering issues related to the software construction that ontology does not cover. This paper offers a proposed solution to that problem, including conceptual aspects, and necessities methods to model commercial Multi-Agent Systems, using fuzzy logic and evolutionary algorithms.

1. Introduction

AI technologies that can exploit ontologies through artificial reasoning are beginning to appear at the heart of e-commerce platforms. A big effort is being made to generalize ontologies, in order to make more powerful e-commerce systems. The main purpose of ontology is to enable communication between computer systems making it independent from the individual system technologies, information architectures and application domain. In agent based systems, the adoption of a shared ontology allows commerce agents to simultaneously interoperate without misunderstanding and retain a high degree of autonomy, flexibility and agility. Beyond ontology, there are some concerns related with this scenario, that involve software engineering problems, and business model problems, that can be summarised as:

- The complexity of the ontology of a marketplace can exceed that of the largest and most sophisticated knowledge based systems.
- Products and services need to be represented with sufficient richness that can be understood by all of the different viewpoints within industry (e.g. the consumer, manufacturer, distributor, etc.).

- The system can offer interoperability, between the business systems of the net market maker and their trading participants (agents).
- The system must be scalable, to cope with the ever increasing number of market participants and to support the integration of new participants.

In essence, ontology can solve the vocabulary problem, and *must be embedded in a software system that can identify potential customers, pre-select them, and serve them using a criterion based on particular business rules.*

To meet all these requirements *minimising computational complexity, server resource consumption* and lack of generalisation, a new perspective is necessary to design and to modify e-commerce multi-agent systems [4, 7, 9]. This paper proposes a method that includes a model and an architecture, which is based in computational techniques that are successfully used to solve similar problems.

2. Scope and Domain Definition

The proposed method can be used in a broad range of multi-agent systems, but this paper will focus on the requirements of e-commerce systems, because that platforms follows a business rule based model strongly based in ontologies. For as the information economy grows in significance in Europe and the USA, the experts predict that ontologies will become increasingly significant to both new e-commerce start-ups and long established businesses [7]. This means that there is a big opportunity to Artificial Intelligence, especially multi-agent systems, to do a jump in commercial platforms as the object of this work. Fuzzy Evolutionary Multi-Agent System (FEMAS) was trying to achieve three objectives:

- To enhance and improve any existent e-commerce system, from simple CGI applications, to more sophisticated platforms. This is called *provider system*.
- To allow the provider system a more realistic and efficient use of computational resources.
- To unify ontologies with business rules in a simple, robust and scalable model.

3. FEMAS Conceptualisation

3.1. Conceptual and Foundational Aspects

FEMAS is the acronym for Fuzzy Evolutionary Multi-Agent System, it comprises a conceptual model, an architectural design, and a method to help building commercial multi-agent systems. FEMAS is strongly based on the concepts and methods used to build control fuzzy inference systems, because the problem addressed here can be seen as an automatic control problem. The likeliness are based on the fact that agents as clients, can be seen as input variables that must be controlled by the provider services, using business rules. Then the output control

variable is a measure of agent preference degree for some product or service offered.

Moreover, the evolutionary aspects depicted in FEMAS come from the ALife research field, especially from results of the Tierra project [14, 15], which to date was the mayor effort to simulate evolutionary conditions on computer processes. It has a very simple and powerful design of Virtual Operating System (VOS), used to apply evolutionary pressure over the organisms, and to control computational resource use. This VOS model is useful in B2B processes, because the customers can be seen as organisms that can fit in some virtual business world rule by specific business rules. The scheduler is responsible for two main tasks: to *select* an organism (or agent), and call the *reaper* when the agent population is threatened by lack of service quality or increasing resource consumption (server collapse). As side effect, this kind of scheduler is more realistic, from the marketplace point of view, and uses fewer resources than a conventional FIFO (First In First Out), broadly used in software systems and in operating systems as UNIX or Windows [16, 17].

FEMAS integrates fuzzy control systems and evolutionary systems, to construct a more realistic customer/provider architectural model that can be used to create or to enhance real-world commercial multi-agent systems. The FEMAS foundational concepts are:

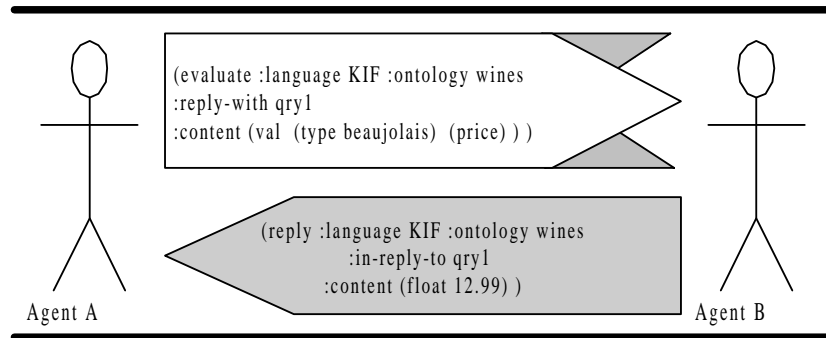
- analogy between rule driven soft agents and the representation and management of subjective knowledge which represents linguistic information such as expert information in fuzzy control systems, which means agents that act as sensors providing input values,
- isomorphism between ontology and fuzzy logic linguistic variables,
- operating functionality shared by net market process and computational evolutionary systems containers, where the rule selection by fitness is the optimal scheduling strategy.

3.1.1. Ontologies, Fuzzy Linguistic Variables and Business Rules

Ontology is a formal explicit description of concepts in a domain of discourse [3]. Concepts, sometimes called classes, are the focus of most ontologies. Classes describe concepts in the domain, and have slots, which describe properties of classes. Developing an ontology includes: defining classes, defining slots, describing allowed values for these slots, and filling in the values for slots for instances [2, 12].

Ontologies are made to agree about the universe of discourse within specific knowledge exchange, and to require a specific related service. KQML (Knowledge Query and Manipulation Language), provides two keywords to allow agents to use ontologies [12]. The keyword *:ontology* which is the term definition used in the *:content* message parameter, and the keyword *:content*, which indicates the information about which the message expresses a requirement. Sample 1, a simple KQML conversation involving ontology can illustrate this point.

The *:content* parameter, is expressed following the universe of discourse allowed for the predefined *:ontology* name, in other words the *:ontology* parameter implicitly defines the allowed values for *:content* parameter, thus KQML provides a context free mechanism to broker in multi-agent systems, using any type of slot in the *:content* parameter with a previous agreement on *:ontology* name meaning. In a similar way, fuzzy control systems use linguistic variables to represent the



Sample 1 - KQML conversation including ontology.

ontology of the problem and this ontology determines the universe of discourse associated to the linguistic variable. The main difference between both uses of ontology is that in fuzzy systems the ontology is implicit in the controller's function while in agent environments the ontology is explicit by name. The only requirement to satisfy to apply fuzzy control techniques to multi-agent environments is to make the ontology explicit.

Fuzzy inference systems used in automatic control are generally embedded in the controller hardware, which receives inputs from sensors as integer or real numbers and produces as output another number which correspond to some control action. A pressure controller has an implicit ontology: *pressure*. This ontology is called linguistic variable, and the input range of sensor is named the universe of discourse and symbolized by the capital letter **U** [5, 8].

A linguistic variable has also a set of terms, which cover the entire universe of discourse. For example, a pressure sensor which produce inputs within a range of 100 to 2300 psi (pounds per square inch), can be described as follows:

$$T(\text{pressure}) = \{\text{low, medium, high}\} ; U = [100, 2300] ; \text{input } x/x \in U.$$

The terms in a fuzzy control system are used to map numeric input to a membership degree of a concept named (ontologically) by the term. In addition, fuzzy controllers are described by means of some linguistic control rule sets. One of the most popular types of rules is used by FEMAS:

$$\text{IF } x_1 \text{ is } L_j \text{ and } x_2 \text{ is } L_k \text{ and } \dots x_p \text{ is } L_q \text{ THEN control action is } u_i. \quad (1)$$

Where i is number of control rule, N is number of rules, L_j, L_k, \dots, L_q are linguistic values of the controller's input variables x_j, \dots, x_p respectively and u_i is crisp control action for i -th rule. The result of the controller's action is a crisp value u that is a composition of all fuzzy rules with some compositional rule of inference and defuzzification. Usually, max-min composition and center of gravity defuzzification, the crisp value is obtained as follows:

$$u = \frac{\sum_{i=1}^N w_i u_i}{\sum_{i=1}^N w_i} \quad w_r = \min_{i=1}^p \mu_{L_k}(x_i), r = \frac{1}{N} \quad (2)$$

Here μ denotes the membership grade of corresponding inputs of controller (x_j, \dots, x_p) with respect to given linguistic values L_j, \dots, L_q and w_i is the degree in which i -th rule influences controller output. This kind of fuzzy inference system is called *Sugeno model*, and is useful because it is easy and inexpensive to compute [5]. FEMAS adopts this model to obtain a value that can be used as membership degree to business model, expressed by business rules.

Business rule applications can be found in any business domain that enforces dynamic and frequently changing statements of business policy in application code. A Major e-commerce platform, such as ILOG, provides they software with business rule mechanisms [19]. The rule forms are identical to control rules shown in (1), e.g.:

$$\text{IF inventory is available and client is frequent THEN serve (ASAP)} \quad (3)$$

So, if any business rule can be expressed as a control rule, then is possible define a *broker*, which is a controller, capable to score an agent. This score value, called *schedscore* is the result of defuzzification process shown in (2), where w_r value is obtained from the expressions like *serve (ASAP)* shown in (3). This expression represents a linear function related to quality service measure or transaction timeout restriction. Thus, *schedscore* can be used to reduce the average time of transaction, or suggest some additional offer, based on the business target. Table 1 depicts the relationship between fuzzy control terminology and FEMAS terminology, and correlates the main concepts explained in this section.

Table 1 - Relationship between FEMAS terminology and Control Fuzzy Logic Systems terminology.

FEMAS		FUZZY LOGIC SYSTEM
Business Rules	_____	Control Rules
Brokers	_____	Controllers
Ontology	_____	Linguistic Variable

3.2. Architectural Design

FEMAS accepts KQML keyword pairs *:ontology* and *:contents*, as primary communication with the agent. Each pair of keywords is named a *characteristic* of the agent. Agent characteristics can be configured using forms or templates, it depends on if the agent was created interactively by asking a user, or automatically created by an information system. In both cases the only prerequisite is an agreement about the ontology (by name) and the universe of discourse, expressed as a number. FEMAS architecture is a layered architecture consisting of a *selector*, a *broker*, and a *scheduler*.

The agent first communicates with the *selector*, which is responsible for the agreement on business domain. After that, the agent is assigned to the *broker*, which applies the corresponding business rules on agent. Finally, the result of this process (the *schedscore*) is used by the scheduler to select the agent to be served by the provider services.

FEMAS broker system architecture is depicted in Figure 1.

Selector Process

The first layer the FEMAS architecture defines only receives those agents that can fit with the business target. This is the main function of the *selector*, to accept or reject agent requests based upon agent *characteristics*. When an agent initiates a communication with the *selector*, it looks up in the parameters database searching for an ontology name. If the search not successful the agent is rejected, else the

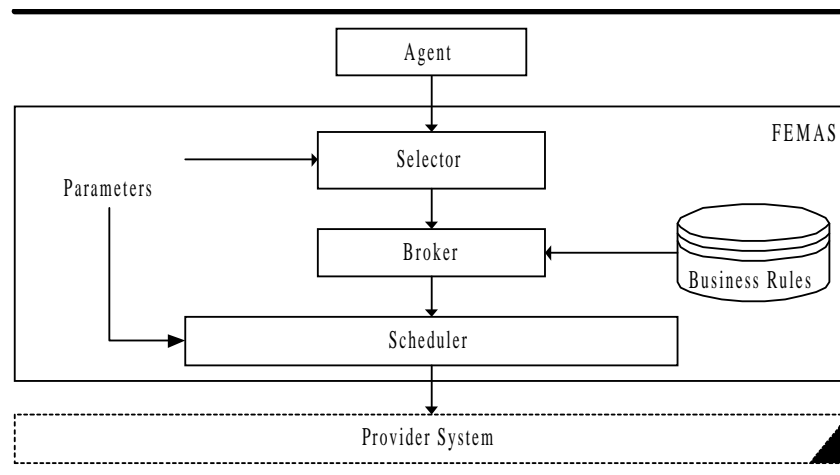


Figure 1 – FEMAS Layered Architecture.

:content value is examined and test for fit in the ontology universe of discourse . If the test is successful the agent is accepted, otherwise is rejected.

The main advantage in using a parametric universe of discourse is to allow the selector to differentiate businesses in the same marketplace, because the U of the

same ontology can differ from one system to other. This differentiation is made without loss of information, and without changes in agent semantics. The selector process can be summarised as follow: if the selector can't find the ontology name or the value associated doesn't fit in **U**, then the agent is rejected. Otherwise the agent is accepted and assigned to a broker.

Broker Process

The broker is a controller; it is a set of business rules instantiated using an ontology name. The broker processes all rules associated to the ontology and respective universe of discourse and obtains a value. This value, which is the aggregated conclusion of the rule, is associated with a service or product. Because the rules are expressed in terms which mimic the linguistic knowledge about the business and the aggregation process is, basically, a scoring process that guarantees the agent satisfies in some degree the expectations about the target. The result value produced by the broker is the *schedscore*, and can be seen as the degree of interest from the business perspective to the target agent. There is one broker for each product/service scheduled.

Scheduling Process

Evolutionary process are based on a fitness concept, the fitness can be defined as

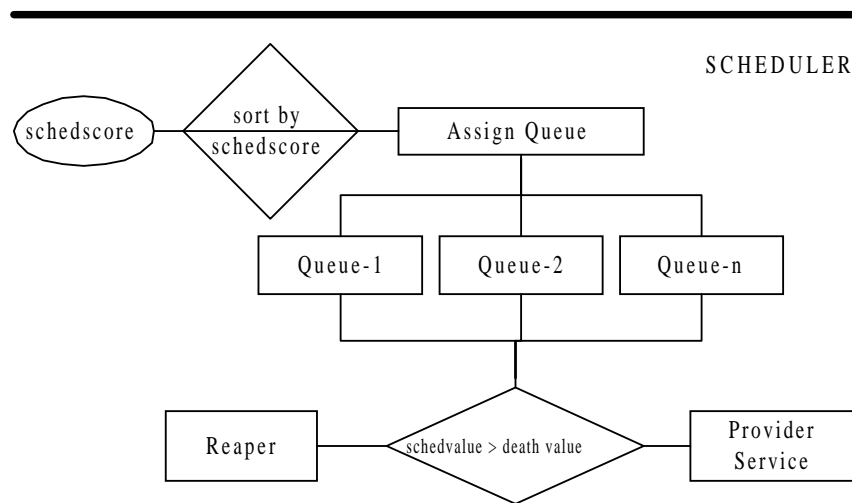


Figure 2 – Scheduler Process.

“... an assumed property of a system that determines the probability that that system will be selected.” [14, 18]. This probability is usually calculated using some population statistic technique, but if this value is calculates using a fuzzy

technique such as here, the probability is replaced by a possibility which is more precise, because it is represented by degree not relative frequency [6]. The *schedscore* is an explicit value that represents a degree of fitness to the business model. Thus, the scheduler can use this value to maintain an ordered queue by product or service using this value as an intrinsic fitness indicator. If the resources are low, a reaper routine can be called by the scheduler in order to free agents that fit over a parameterised value (death value). Once the scheduler selects the agent, it is passed to a provider service, which serves the agent. The multi queue partition can allow intrinsic serialisation or parallelism depending on the provider architecture. When the agent is dismissed, the scheduler removes any entry of the agent in the queues. The scheduler process is depicted in Figure 2. In short, the entire broker process defined by FEMAS can be summarised as follow:

- Accept or reject the agent depending on the ontology business domain.
- All the rules associated with ontology are fired, and the *schedscore* is obtained.
- The scheduler iterates, sending in each cycle, in sequential or parallel mode, those agents that are in conditions to be served by the provider according to best fitness to business rules.

4. Method

The method proposed here could be taken as a checklist necessary to develop and implement a FEMAS architecture. Depending on platform and languages used to implement the multi-agent system, other steps can be taken or some steps can be extended, but at least the following steps must be made.

4.1. Define the Ontology

The ontologies can be defined or adopted from some standard repository. There are some repositories with standard ontologies that can be used [4, 20]. The ontology must be public, in order to allow agents to use it via KQML. Each term in ontology will be a condition in business rules.

4.2. Define the Universe of Discourse

For each ontology is necessary to define the range of associated values that is the universe of discourse; these values can be discrete or continuous. When the values are discrete, it will be modelled as series. When the values are continuous, it will be model as an integer or real range.

4.3. Define the Business Rules

Express the all business rules in the form:

If condition-1 [and / or condition-2 [, and / or condition-n]] then take some action

Where the conditions are logic relationships between ontology terms and *take some action* is a linear function associated to service timeout, service quality measure, product ranking or any measure useful for selecting and scheduling clients.

4.4. Define the Products and Services

For each product or service various tasks must be done:

- Define the business rules associated, this is the *broker* for the product or service. In other words, create the broker.
- Link the products or services with the software system. This can be done creating a software relationship between a product/service and a provider service. The scheduler that sends the agent to the provider service will use this relationship.
- Define a scheduler queue.

4.5. Define the Membership Functions

For ontologies define the shape of the membership functions associated to each term. Their shape depends on the significance or interpretation of the term. This can be done keeping in mind the fuzzy inference engine will be used.

4.6. Create the Brokers

Create a controller or broker based on 4.4, keeping in mind that a broker can share rules with other brokers, this step is closely related with the fuzzy inference engine that will be used.

4.7. Define Operational Parameters

The scheduler and the selector need a parameter database in order to make selections. These parameters are very important because they impact directly over the resource use and overall performance of the system. The basic values to define are death value and the ontology set defined in 4.2. The death value is used by the reaper routine to force dismissal of agents that exceed this value, which not only frees agent resources also terminates the life of the agent in some ordered and logged way. The selector as described in 3.2 uses the ontology set.

5. Conclusions and Future Work

There are no doubts about the need of more intelligent software actors in the e-commerce business software, but the cost and complexity of the solutions produces a dropdown in industry. From this point of view FEMAS is a first step to make more cost effective and reliable hybrid systems. This architecture is easy to implement, doesn't impose performance penalties to the server systems and can (depending on the complexity of the rules model) provide a differentiation factor at the same time that helps to standardise e-commerce ontology efforts. The author

is working on a second-generation FEMAS architecture that covers problems associated with standardised FEMAS as a plug-in server, and creates a new layer of federated *brokers*. The objective of this is to allow installed Web servers a plug-in FEMAS to achieve a balanced load of resources, and to enhance the FEMAS architecture in order to provide several abstraction levels of scheduling capable of operating in distributed operating environments.

References

- [1] Iglesias Fernandez. C. "Fundamentos de los Agentes Inteligentes." Informe Técnico UPM/DIT/GSI 16/97 (Spanish).
- [2] Noy N. F. and McGuinness D. L. "Ontology Development 101: A Guide to Creating Your First Ontology." Stanford University.
- [3] Gruber T. R. "A Translation Approach to Portable Ontologies." Knowledge Acquisition, 5(2):199-220, 1993.
- [4] Hendler J. "Agents and the Semantic Web." pp. 30-37 IEEE Intelligent Systems Journal.
- [5] Mendel J. "Fuzzy Logic Systems for Engineering: A Tutorial." pp. 345-377 Proceedings of the IEEE, Vol. 83, No. 3.
- [6] Kosko B. "Fuzziness versus Probability." pp. 211-240 Int. J. General Syst., Vol. 17.
- [7] Smith H. "The Role of Ontological Engineering in B2B Net Markets." URL: <http://www.ontology.org/main/papers/csc-ont-eng.html>.
- [8] Roger Jang J. "Neuro-Fuzzy Modeling Control." pp. 378-406 Proceedings of the IEEE, Vol. 83, No. 3.
- [9] Smith H. "Information Architecture for Net Markets ... or how to thrive and survive in the B2B cyberspace." First European Conference on eMarkets, May 10th-11th 2000, Stockholm, Swedish Trade Council.
- [11] Fensel D. "An Ontology-based Broker: Making Problem-Solving Method Reuse Work." Proceedings of the Workshop on Problem-Solving Methods for Knowledge-based Systems at the 15th International Joint Conference on AI (IJCAI-97).
- [12] Finin T. et al. "Specification of the KQML Agent-Communication Language - plus example agent policies and architectures." URL: <http://www.cs.umbc.edu/kqml/papers/kqmlspec.ps>.
- [13] Gruber T. R. "Toward principles for the design of ontologies used for knowledge sharing." International Journal of Human-Computer Studies, 43(5,6):907-928.
- [14] Ray T. "Evolution complexity entropy and artificial reality." Physica 75:239-263.
- [15] Ray T. "Documentation for the Tierra Simulator." Tierra Simulator V4.2.
- [16] Tanenbaum A. Woodhull A. "Operating systems." Prentice Hall.
- [17] Tanenbaum A. "Distributed Operating systems." Prentice Hall.
- [18] Heylighen F. "Fitness." URL: <http://pespmc1.vub.ac.be/fitness.html>.
- [19] ILOG Rules Products. URL: <http://www.ilog.com>.
- [20] The DARPA Agent Markup Language (DAML) Program. URL: <http://www.daml.org>.