METAOSE: Meta-analysis for agent oriented software engineering

Luis Razo[1,2], Félix Ramos[1]
Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional[1]
Av. Científica 1145 , colonia el Bajío, Zapopan , 45015, Jalisco, México.
lrazo@gdl.cinvestav.mx, framos@gdl.cinvestav.mx

Michel Occello[2]
University of Grenoble[2]
Laboratoire de Conception et d'Intégration des Systèmes
50, rue Barthélémy de Laffemas, 26902, Valence, France
michel.occello@iut-valence.fr

Abstract

In view of the existing methodologies for agent-oriented software engineering (AOSE) the development of multi-agent systems (MAS) is still a difficult challenge. The learning curve for mastering MAS model properties and problem's domain characterization is steep. The software engineers hesitate to use MAS since choosing a MAS-based methodology leads to fix the type of models that will be involved without inter-methodologies reusability. We think these are some of the reasons restricting the dissemination of multi-agent methods. This paper presents a self-organized MAS-based intelligent process to assist the engineer. This process comprises three stages: problem features and domain characterization, MAS components matching and meta-analysis. It aims to reduce the difficulties of starting an MAS-based solution to disseminate the use of MAS. The process is presented as a guiding tool for the engineers, especially for those less experienced in MAS, creating a path in the preliminary stage of the MAS conception. This approach is situated just before choosing a MAS framework or methodology to deploy a solution. We show how it works and we present a study case to compare our preliminary results.

1. Introduction

Taking into account the progressive evolution of technologies, systems will comprise each time more distributed devices capable to host software. The hardware and software dual mixture piece will propose functions as individual unit or within large complex systems increasingly autonomous and communicative, via cable or wireless. These next generation applications will deal with varied fields and users concerning diverse sort of higher and cognitive abilities where the applications will need to merge reasoning competences. Consequently they will use intensive communication and highly developed organization; also, the next generation applications will need to adapt themselves to users' profiles or abilities. Therefore they will perform all these tasks without forgiving to address security. These future evolutions of applications oblige to craft a rupture in the current approach utilized to analyze and design them. A relevant response is the use of life simile to model such applications. It means to use the biological or social similes to create a new analysis and design process. To adapt the processes of development, we consider essential to make certain a real option by suggesting solutions which integrate these similes in applications. The multi-agent paradigm is an approach among these using such similes.

A great variety of methodologies multi-agents oriented can be found today allowing the software development conduction by means of multi-agent models (for a MAS survey see [2]). So the variety of methodologies adds the making decisions difficult foremost for the non-experienced engineers. On the other hand, software engineers are indecisive to use MAS since choosing one method that could carry out to attach the type of models involved. In sequence when the designers are choosing a method they must master the multi-agent model properties and also their must know how to match them to the application's features and domain. In cir-

978-0-7695-4204-1/10 $26.00 © 2010 IEEE
DOI 10.1109/CERMA.2010.33

cumstances of software production a common problem for the engineers is the solution's domain misunderstanding [9] because the engineers are not familiar with all the domains where they have to work for a solution; usually they must identify the domain and consequently learn the domain's background to manage with the development process. Nevertheless the MAS-based solutions have demonstrated to be a means that successfully helps to solve complex problems in different fields.
The primary intent of our approach is to disseminate the use of MAS in the industry by means of simplifying the access to it. Basically our aim is to guide the developer through the MAS construction in order to make this alternative a viable and trustable solution.

2. Approach overview

2.1 Issues and proposed solutions

We have identified a set of main lacks within the MAS development analysis phase process. So for each issue we propose a solution within the approach context:

2.1.1 *"Problem characterization - domain of solution" relation*

- Issue: When an engineer choose a methodology to solve a problem using MAS he must first identify the desired application problem's characteristics and domain. Also he must be related with the problem's domain. This information is useful to choose a compatible methodology that provides accurate support for the problem's domain. Nevertheless an engineer's common difficult is the domain misunderstanding [9] in the software engineering context. That's why is difficult to characterize an application specification into problem features and domain specification targeting a MAS-based solution.

- Solution: The relation "problem characterization - domain of solution" make us realize that we need to characterize the application specification in the MAS context before start choosing a MAS methodology. The aim of this characterization is find the problem's features and domain's specification, thus, relate them to the domain of solution of the available methodologies. The application specification is provided by the user (software engineer, developer, etc.), it is a software engineering based text that contains: entire problem context description, problem keyword set, target field description, domain glossary of terms, etc.

* Application Specification: We propose to identify the problem features and domain specification parsing the application specification document[18]. Thus the application specification is treated by characterization engine that analyze it with the objective of typify the application finding problem features and domain specifications. To achieve it, the characterization engine works using three algorithms: text analysis algorithm, domain specification locator and characterized data serialization. The application specification is composed by a text structures formalized in the next sets formalizations: The word set is $W \equiv \{\omega_1, \ldots, \omega_w\} \forall_k \in [1, w]$, where each ω_k is a valid word found in the application description. The sentence set is $S \equiv \{\sigma_1, \ldots, \sigma_s\} \forall_k \in [1, s] : \sigma_k \subset W$. The paragraphs set is $G \equiv \{\rho_1, \ldots, \rho_g\} \forall_k \in [1, g] : \rho_k \subset S$

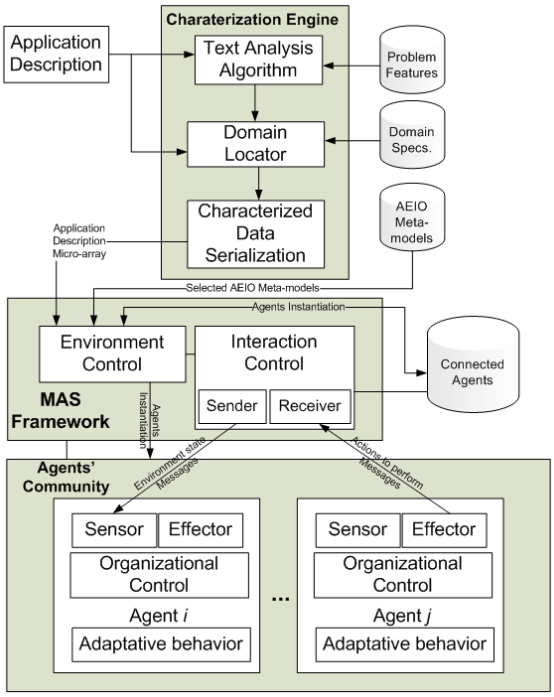

Figure 1. Overall Architecture

* Problem features and domain knowledge base [9]: The search for the problem characteristics and domain context leans into a knowledge base structure composed by problem features. These features are composed by textual structures paragraphs, sentences and finally keywords previously identified within one or more domains and uses a *belonging-to-domain* probability measure. Thus the problem features formalization starts with a set of textual description of problem feature: $T \equiv \{\tau_1, \ldots, \tau_t\} \forall_i \in [1, t]$ Where each τ_i is a feature description abstraction. Then we define a problem feature text structure set as follows $K \equiv \{\kappa_1, \ldots, \kappa_m\} \forall_k \in [1, m]$ where each κ_k is a text structure (a keyword, sentence

or paragraph) related to one or more problem features. Finally the set of real values where each one represents a result of the application specification text analysis as a belonging probability for each related text structure $V \equiv \{\nu_1, \ldots, \nu_v\} \forall_k \in [1, v] \, \nu_k \in \Re, [0, 1]$

Finally the set of real values where each one represents a result of the application specification text analysis as a belonging probability for each related text structure: $V \equiv \{\nu_1, \ldots, \nu_v\} \forall_k \in [1, v] \, \nu_k \in \Re, [0, 1]$ The next part is to define the domain specifications. To do this we must define a problem features belonging probability as a set of real values: $B \equiv [\beta_1, \ldots, \beta_b] \forall_k \in [1, b] : \beta_k \in \Re : \forall_{\beta_k} \in [0, 1]$ Also the set of domain structure representations $R \equiv [\psi_1, \ldots, \psi_b] \forall_k \in [1, b]$ where each ψ_k is a text structure that represents partially a domain. Also we define a domain as follows $D \equiv [\delta_1, \ldots, \delta_d] \forall_k \in [1, d]$ where each δ_k is a domain that is represented by a domain structure composed by subsets of R. Therefore we define the set of problems $P \equiv \{\gamma_1, \ldots, \gamma_p\} \forall_k \in [1, p] : \gamma_k \subset R \times D \times B$. Therefore the micro-array characterization set is defined by the next relation: $C \equiv [\theta_1, \ldots, \theta_c] \forall_k \in [1, c] : \theta_k \subset P \wedge F$

* Treat with uncertainty [5]: Throughout the text analysis an evaluation with the problem features knowledge base is performed. Then a belonging domain probability for a problem feature is calculated using the results. The aim of this is to discover the different domains belonging probabilities in relation with the identified problem features domains words that have appeared. So, then this part is related to the text metrics and text data mining that is not the focus of this publication.

* Learning: How to enrich, improve and update the problem features and domain knowledge base has is origin in an automated learning [20]. This learning is based in the general ontology domains analysis together with the problem and domains specifications provided by the user. But in this publication we do not focus on this aspect.

2.1.2 *Compatibility and reusability features*

- Issue: Looking at the diversity of options on the existing MAS methodologies we quickly realize that there exist a rich variety of methodologies [2]. Each of these methodologies uses different process, models and components. This variety can confuse the developer since most of the methodologies do not explicitly detail their overall solution domains. Subsequently we found that is difficult to know the methodology overall domains without prior experience about which kind of situations is best dominated for which methodology as is common with non-MAS software development [9]. From global to local also is hardly to know the belonging domain of each individual development process, model or component within a methodology. So, the methodologies are not standardized between them at all, in other words is not easy to use shared features from two or more different methodologies. This is because we cannot profit different pieces from different methodologies to build a MAS based solution also we have not information about the belonging domain of each individual methodology piece at local level.

- Solution: These issues make us conceive a solution based on the Model Driven Engineering (MDE) approach with the aim of standardizes the reusability and compatibility between methodologies features. This allows us to abstract all the features into meta-models that will represent all the MAS aspects from the methodologies but taking advantage from the patterning approach similar as is proposed in [17]. It will allow us to profit the reusability of advantages from different methodologies sources. Thus it provides us accurate information about the compatibility restrictions between the meta-models. So, the MDE has proven be a successful tool in the software engineering area and widely accepted in the industry [10]. Nevertheless it must be adapted to work within the MAS approach using a reliable way to divide a MAS approach into meta-models. In consequence we have based this meta-model decomposition on the MAS vowels components proposed by [8] and best known as AEIO approach. So at this point we provide a simple way to create compatibility and reusable MAS based meta-models approach. To formalize the metamodels first we need to define the meta-models types set: $H \equiv \{A \wedge E \wedge I \wedge O \wedge U\}$ where each type is composed by a set of possible sub-types of meta-model, for agents: $A \equiv \{\alpha_1, \ldots, \alpha_a\} \forall_k \in [1, a]$ Environments: $E \equiv \{\epsilon_1, \ldots, \epsilon_e\} \forall_k \in [1, e]$ Interactions: $I \equiv \{\iota_1, \ldots, \iota_i\} \forall_k \in [1, i]$ Organizations: $O \equiv \{o_1, \ldots, o_o\} \forall_k \in [1, o]$ and unification restrictions: $U \equiv \{\upsilon_1, \ldots, \upsilon_u\} \forall_k \in [1, u]$ Sub-types of meta-model are α_k, ϵ_k, ι_k, $\vee o_k$" or υ_k. Five different types of meta-models and also different sub-types. For example the meta-model type of agent could be a sub-type of dialogical agent, cognitive agent, etc. The meta-models characteristics set: $J \equiv \{\xi_1, \ldots, \xi_j\} \forall_k \in [1, j]$ where each ξ_k is a meta-model characteristic abstraction. For example we can say that we have an emotional characteristic existing in an agent meta-model. The meta-models belonging domain probability values: $Z \equiv [\zeta_1, \ldots, \zeta_z] \forall_k \in [1, z] : \zeta_k \in \Re : \forall_{\zeta_k} \in [0, 1]$ Each meta-model has adjusted values related with one or more domains through their characteristics. We can say that an emotional characteristic is related to an abstraction of a simulation domain type that require adjusting to them quickly with a high value nearest to the maximum one. Thus we can define the meta-models set:

$M \equiv \{\mu_1, \ldots, \mu_m\} \forall_k \in [1, m] : \mu_k \subset H \times J \times D \times Z$. The meta-models must be analyzed evaluating all their features and situate them into domains of solution, thus characterized. So, we start analyzing them like a description composed by an ensemble of features where we analyze from local to global. The result of this is that the AEIO meta-model is stored in a knowledge base together with their characterization value.

So, the final result a dynamically improved AEIO meta-models characterization knowledge base. Thus it brings us the information about within which domain and specific AEIO meta-model can be considered efficient. Using this information we can find a benchmark that allows connecting a given problem features set with a domain or domains where an ensemble of AEIO meta-models are considered efficient. Consequently it also helps to evaluate each AEIO meta-model from global to local thus it make possible reuse them within the domains where they are considered efficient accurately.

2.1.3 *Matching problem characteristics and meta-models domains*

- Issue: Based on the wisdom of both knowledge bases, problem features and domain characterization together with the AEIO meta-models domains characterization, the next issue to solve is how to match these problem characteristics and domain with an ensemble of AEIO meta-models? So, we need to find a way to compare them and predict which could be the best combination of AEIO meta-models to use as a solution for the given problem and thus choose these accurate AEIO meta-models.
- Solution: We consider that a problem is composed by features where each one must be identified through a text or abstract analysis and then get linked at least with one belonging domain. Thus, we propose characterize the problem and domain as well as the AEIO meta-models domain using a probabilistic notation based on the Bayesian programming style [5], the reason is that the probabilistic representation has proven to be a robust alternative to treat with uncertainty reducing the entropy and giving a base to implement automatic learning mechanisms to improve the belonging domain values of the AEIO meta-models and problem features, also is important to mention that the meta-analysis process is exactly based on this kind of probabilistic representation, thus, it is an accurate choose to implement the meta-analysis step too. But for the matching process we propose a MAS with an emergent matching behavior, it means that we define a simple induction algorithm for each agent behavior and self organizing themselves they form groups of possible solutions using the information given by the micro-array probabilistic representation of both parts problem features and domain and AEIO meta-models domain. So, the entire problem features and domain information is coded into a micro-array structure filled by the probabilistic information of the given problem or the AEIO meta-models domain. The micro-array data structure that we use within the matching algorithm is based into the micro-array of genetic information used within MAMA's work [23]. These micro-array representations are used with the purpose of make them machine readable and upgradeable.

2.1.4 *Meta-analysis*

- Issue: The AEIO meta-models combinations obtained through the matching process as groups of agents represent are greater than one and therefore it is necessary to know which of these groups represent the best possible solution.
- Solution: Then we need to make an "analyze of the analysis" of the ensemble of groups results to find the most accurate result to apply for the given problem.[1]. We reason about meta-knowledge features from the analysis of our experience and literature to identify appropriate solutions in terms of meta-models. We consider this approach original in every point in a phase of computer analysis. To do this we use the overall certainty probabilistic value of each group.

2.2 Summary

The present approach is a preliminary MAS development step situated as part of the analysis phase just between the requirements gathering and the design phases that aims to reduce the difficulties of MAS conception providing specific solutions over the common issues. There are several methodologies and platforms that support the design and implementation of MAS. Some of them try to simplify the conception and the design of MAS. In some cases these approaches adopt a specific model or a guide for development. Furthermore, the platforms usually focus on the problems covered by their domain, applying a method composed of models to design MAS. The most part of existing multi-agent methods usually divide the multi-agents development cycle in two levels of abstraction: analysis and implementation. Thus, a MAS software engineering development cycle is composed by these two levels of abstraction or main phases. Our approach focuses on the analysis phase. In turn our approach consists of three main inner stages: application specification characterization, meta-model comparison

[1]The term "analyze of the analysis"' is also known as Meta-analysis [16]. The meta-analysis is defined by the National (U.S.) Library of Medicine as *"a quantitative method for combining results of independent studies (usually drawn from published literature) and synthesizing summaries and conclusions that can be used to assess therapeutic effectiveness, plan new studies, etc., with application mainly in the areas of research and medicine. "* For us, it is therefore equate *"clinical studies"* to *"study problems domains"*

and meta-analysis. All the process begins with the application specification provided by the software engineering requirements acquisition process. It describes the desired application from where we must abstract the problem features and locate the domain specification. So, any application specification can be divided in two parts: problem features and domain specifications. Both parts bring the first hints to be conceiving a MAS-based solution. In our approach we propose to use a textual version of an application specification. Thus automate the analysis through a text analysis process to characterize the textual content producing a problem features and domain characterization. Therefore encode the results into a micro-array assay. This is followed by the second stage where is performed the meta-models matching process. [2] Therefore the matching process is a benchmarking method that compares the problem features and domain characterization with the available meta-models domain of solution. This is constituted by a multi-agent system where each agent is instantiated as required. Every agent represents one meta-model selected from one of the four meta-models using the unification restrictions. The selection is done as the first part of the process using the application specification micro-array data. The criterion is to select all the meta-models that are using the most part of the domains contained in the application specification micro-array. Therefore when each agent is activated it receives booth micro-arrays their respective meta-model features characterization and the application specification. The agent architecture is composed of the next units:

- Agent organizational control. This control is the organizational part of the entire multi-agent system that corresponds to the agent architecture. It is composed by their self-representation, which is composed by the represented meta-model and the application representation application specification micro-arrays data. The group membership information is registered into the agent's group organization. The third agents' representation is the micro-array data of the community known agents.

- Agent interaction control. This is the interaction part into the agent. It is composed by two communication dedicated modules: Sensors and Effectors. The sensors receive the environment stimuli or messages and the effectors send messages and perform actions.

- Adaptive Behavior. This is fashioned from the proposed decisions of the induction module in set with the negotiation module. The induction module is a module that owns an induction algorithm that must evaluate new group memberships and group creations using agent's self-representation and the already group members agents. In other words this algorithm evaluates if is suitable to create a group with other agent or to attach to an existing agents group. These evaluations are done using a Bayesian probabilistic distribution to evaluate the probability of solution using a certain meta-model μ_j as partial solution for the application description problem θ_i (agent's self representation):

$$P\left(\; Solution \,|\quad \theta_i \wedge \quad \mu_j \; \right)$$

Then to know the certainty of using two different meta-models μ_j and μ_k as part of the solution we use the following Bayesian probabilistic distribution (external agents' representation and group membership evaluations):

$$P\left(\; Solution \,|\quad \theta_i \wedge \quad \mu_j \wedge \quad \mu_k \; \right)$$

As described in [5] these distributions uses a combination of current values (the application description micro-array characterization θ_i) and historic values (meta-models characterizations μj and μ_k). As example we can say:

$$P\left(\begin{array}{c} Solution = true \,|\, type = \alpha_1 \\ \begin{array}{cccccc} features & & domains & & charac. \\ \tau_1 \wedge & \tau_2 \wedge & \delta_1 \wedge & \delta_2 \wedge & \xi_1 \\ false & false & false & false & false \\ false & true & false & true & false \\ true & false & true & false & true \\ true & true & true & true & true \end{array} \end{array}\right) = \left(\begin{array}{c} 0.11 \\ 0.65 \\ 0.55 \\ 0.89 \end{array}\right)$$

So we want to find values that are on the right slope of a gaussian bell curve with the aim of reduce the entropy (see fig. 2). So in the last example case we can see that the combination of α_1, τ_1, τ_2, δ_1, δ_2 and ξ_1 they are a solution with a probability value nearest to one. So, we can say that the meta-model α_1 is as a high promising solution. On the other hand the negotiation module creates and interprets the required messages to negotiate with other agents the group memberships.

- Planning Module. This module controls the actions to perform in the timeline. It keeps the control of the agent using all the modules and controls to create the agent's behavior. The behavior is adaptive, and it is composed by reactive actions and cognitive choices.

The agents interact into the agents' community and the results of these interactions are the creation of agents' groups.

[2] So, a meta-model is a model description or model of models in our case we use a MAS based approach called AEIO MAS [8] that decompose a MAS in four components Agent, Environment, Interaction and Organization.

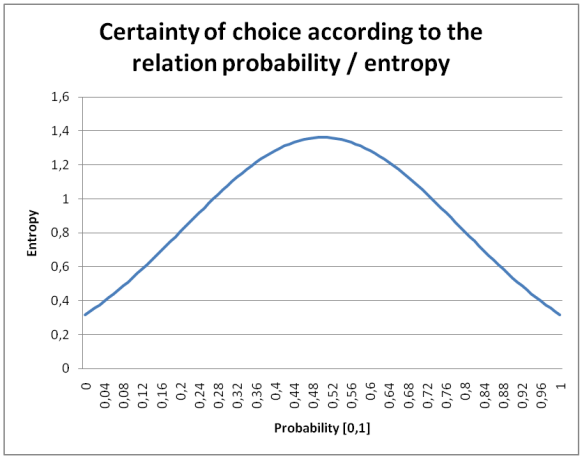

Figure 2. Certainty or entropy distribution related to the probability values.

These groups are conformed using the criteria of the application description micro-array and the individual meta-models micro-array characterization. So, we obtain a list of groups from the agents' community as result. Each group represents a possible solution comprised by the agents represented meta-models and their respective group evaluation value. Thus we have as final result of this step a list of possible solutions where each list member is a possible solution that is constituted by AEIO meta-models that describes an abstract and agentified way to solve the application description.

And as third and final stage the meta-analysis. We understand the concept of meta-analysis as it is defined in medicine [22] but applied to study problem domains. In our approach it consist in evaluate all the resulting groups applying the meta-analysis method described in [16] to choose the most accurate group. The final result is a MAS description based on AEIO meta-models. It means that we obtain an AEIO meta-models set which together all the AEIO meta-models describes an entire MAS application in other words the problem specification is decomposed into AEIO meta-models as an agentified application. These are the meta-models chosen by the matching process and meta-analysis evaluation.

3 CASE STUDY

Applying the approach described in this article, we have developed the next case study using as basis the meta-models characteristics and domain definition as an ontology of micro-array representations.

In our case study we have defined within the domain-problem and meta-models ontologies a small set of characteristics, domain settings and meta-models, which are based on a simple auction problem- briefly described in the following lines.

3.1 Application Description Analysis

The auction is a problem which forms part of the competition and purchase and sale domain [19]. It requires several agents who are customers and try to obtain goods at the lowest price possible. The goods are offered by the auctioneer agent, who in his turn wants to sell them at the highest price. The number of goods is limited, but they are delivered continually. The minimum prices are assigned, which will change according to the customer offers. The customers can offer a higher price for the goods which appear to be more valuable under their inner perception. The prices are increased according to the rules specified in the protocol, which indicates that the new price the customer offers for each article must be higher than the price offered by the previous customer. So, other features are used depending on the primary and secondary types of auctions. Also the minimum rising amount can be established and other customizable features as rules, maximum budget, minimum starting value, etc.

We understand a domain as is defined in [9] *"'A sphere of knowledge, influence, or activity"'*, so, as we mentioned this is the first step and the result is given into a micro-array that contains the information of characterization. To obtain the micro-array content information we search in a similar maner as is done in [21] with the difference that we look into the text to obtain the keywords and map them into the ontology structure to know which is the belonging domain probability.

Using the description of the problem, the key words are selected from the problem description. Then these keywords are matched with the domain-problem ontologies data. For this case we have used only a small set of key words that belong to the defined domains. As already described, the problem features and domain ontology holds micro-arrays that represent a relation between the domain and the problem. Then, we define the micro-array compatibilities for the types of agents, environments, interactions and organizations.

Following the process, the micro-array that describes the problem is applied to query the meta-models ontology. For this case we show a part of the meta-models micro-array ontology with a fragment set of characteristics based on the four types of agent meta-models: reactive, cognitive, hybrid and dialogical ones.

In the next step the meta-model micro-arrays that best match with the problem micro-array are selected and then they interact as agents in a MAS with the aim of finding the best agent meta-model to solve the problem.

3.2 Matching algorithm

Using the micro-array information the ontology of meta-models AEIO is queried. This filtering action chooses the meta-models that are the best candidates to construct the MAS solution. The meta-models candidates contain a micro-array characterization, defining their belonging domain. This step complements the meta-analysis that will be realized after because a set of meta-models candidates will be selected.

A MAS based solution is constructed using each meta-model candidate as an agent allowing self-characterization as a relation matching criteria. Each meta-model agent looks for partners matching its micro-array self-description with the micro-array description of other meta-model agents, considering the micro-array problem characterization using a simplified induction algorithm (we propose a Bayesian-based one similar as the utilized in [5]) with a set of compatibility rules to accomplish that are defined in each component. With this simple behavior the agents organize creating relations. From these relations emerges the solution to the problem characterized in micro-array. So, the solution emerges as different groups.

The admissibility of various groups allows us to obtain several solutions. Then the meta-analysis must be performed to decide, which solution is the most suitable.

3.3 Meta-analysis

This step makes possible to analyse each group as a possible solution; the certitude of each group is evaluated under the probability based rules derived from the set of micro-arrays meta-models characterization. This will help to choose from the set of all the possible solutions the most accurate one. The final result is the set of AEIO meta-models that will construct the MAS, which solves the problem.

So, then analyzing the solution groups for our auction problem the final meta-analysis proposed as a suitable solution the cognitive agent meta-model, a semi-observable environment meta-model, an protocol-based conversational interaction meta-model and a client-server organization meta-model . This solution group has resulted the most appropriate.

4 DISCUSSION

Comparing our results with a similar study described in [1] where a comparison between seven different platforms is achieved using a similar auction situation, we can see that the type of agent used in each platform has features of a BDI agent. This type of agent is linked to the type of cognitive agent has been chosen in our approach. However, our case study uses a limited set of features and not all of them are on all the platforms used in [1] and vice versa, but we can see that our solutions have proven to be nearest to the most appropriate. We believe that these results are promising and we will continue working to improve and increase this approach.

The methodologies frequently base the software engineering application on different models and diagrams derived from UML. Their objective is to develop different and specific MAS. The use of models derived from UML is located in the M1 layer of the four-layer Model Driven Engineering (MDE) [14]. It means that these models belong to a specific meta-model. Each model is designed to solve problems for a specific aspect, such as agent creation or communication.

The models defined in each platform are different, for instance, an agent model defined in PASSI [7] could not be used in ADELFE [4]. But how could we solve a problem that requires a PASSI [7] agent model type, which is running embedded according to DIAMOND [12] specifications? In the next lines we explain how our approach contributes to the solution of this kind of problems using a meta-model definition that belongs to the M2 layer of the four-layer MDE [14].

It has been observed that all the afore-mentioned methodologies provide solutions focused on a specific kind of problems and domains. On the other hand, it is difficult to decide in advance which methodology would be appropriate to solve a specific problem. In other words, the domain covered by each method is limited, for example, ADELFE is geared only to the domain of adaptive multi-agent systems and Gaia is directed to a closed domain with multi-agent static characteristics.

The truth is that we cant solve all the possible problems using a single method, because each method covers a limited domain, in which only problems suitable for this domain can be solved. Therefore, a problem that can be solved with one method efficiently probably cannot be solved with another.

For this reason our approach proposes to establish a preliminary stage, in which the problem is analyzed to determine its domain. This verdict guides the developer stating which would be the best way to solve this problem under the predefined selection criteria. Besides our approach defines the use of meta-models based on the AEIO Decomposition [8] for MAS approach. The aim of these meta-models is the reutilization and adaptation of the different models that already exist, providing an option to make MAS. This aspect is similar to building blocks or design patterns[13], but our approach will neither change the existing models in other methodologies, nor propose a new model standard.

This proposal should not be reduced to only use the MDA approach (Model Driven Architecture) [14] modified

for the MAS engineering. We are not focused on the use of meta-models to transform them into models of implementation in the sense adopted by MetaDIMA [11], ADELFE [4] and PASSI. This technique can be used for detailed analysis, but as already mentioned, we want to work on an preliminary conceptual analysis. Our goal is not to unify the different metamodels into one as proposed in [3]. We can not adopt a single metamodel as in [15]. It is impossible taking into account all in only one metamodel because of the difficulties at the implementation and at the deployment in a different way in terms of domains. Instead, to describe our meta-model (analysis model) we are forced to use a Meta-Meta-model as proposed by the method MESSAGE / AUML [6] (it uses the MOF UML).

5 CONCLUSION

Our approach is a new alternative for the developers to encourage them in the use of MAS based solutions. This approach is positioned as a preliminary phase of software engineering where the system designer can evaluate the MAS as a possible way to follow. The approach described here is the first part of a complex solution that is still under development. Nevertheless it considers important aspects, such as reusability of existent MAS models and update capability of the domain-problem ontology and meta-model ontology. The micro-array profits these upgradable aspects. The solution emerges from matching performed in a MAS, where the meta-models characterization acts as an agent with certain behavior oriented to looking for another type of meta-model agents. Once such agents are found, the matching is performed and relations between them are created. The last part consists in meta-analysis of the set of possible solutions, where the certitude criterion is applied to find the most accurate solution. The final result is an AEIO meta-model set.

References

[1] J.-P. Barthés, P. Chevaillier, R. Courdier, Z. Guessoum, O. Gutknecht, P. Mathieu, and M. Occello. *Organisation et applications des SMA*. Hermes science publications, 2002.

[2] F. Bergenti, M.-P. Gleizes, and F. Zambonelli, editors. *Methodologies and Software Engineering for Agent Systems*. Springer, 2004.

[3] C. Bernon, M. Cossentino, M.-P. Gleizes, P. Turci, and F. Zambonelli. A study of some multi-agent meta-models. *Fifth International Workshop on Agent-Oriented Software Engineering (AOSE'04) at AAMAS'04*, pages 62–77, 2004.

[4] C. Bernon, M.-P. Gleizes, S. Peyruqueou, and G. Picard. Adelfe: a methodology for adaptive multi-agent systems engineering. pages 156–169, 2003.

[5] P. Bessière, C. Laugier, and R. Siegwart. *Probabilistic Reasoning and Decision Making in Sensory-Motor Systems*. Springer-Verlag, 2008.

[6] G. Caire, W. Coulier, F. J. Garijo, J. Gomez, J. Pavón, F. Leal, P. Chainho, P. E. Kearney, J. Stark, R. Evans, and P. Massonet. Agent oriented analysis using message/uml. In *AOSE '01: Revised Papers and Invited Contributions from the Second International Workshop on Agent-Oriented Software Engineering II*, pages 119–135, London, UK, 2002. Springer-Verlag.

[7] M. Cossentino. From requirements to code with the passi methodology. *Agent-Oriented Methodologies*, pages pp.79–106, 2005.

[8] Y. Demazeau. *Rapport d'Habilitation a Diriger des Recherches- Agentes Voyelles*. PhD thesis, Institut National Polytechnique de Grenoble, Laboratoire Leibniz, 2001.

[9] E. Evans. *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Addison Wesley, 2003.

[10] D. Gasevic, D. Djuric, and V. Devedzic. *Model Driven Engineering and Ontology Development*. Springer-Verlag, 2009.

[11] Z. Guessoum and T. Jarraya. Meta-models and model-driven architectures. *Contribution to the AOSE TFG AgentLink3 meeting*, 2005.

[12] J.-P. Jamont and M. Occello. Designing embedded collective systems: The diamond multiagent method. pages 91–94, 2007.

[13] J.Holland. *Hidden Order: How Adaptation Builds Complexity*. Helix Books, 1995.

[14] A. Kleppe, J. Warmer, and W. Bast. *MDA Explained: The Model Driven Architecture: Practice and promise*. Addison Wesley, 2003.

[15] H. Knublauch and T. Rose. *Tool-supported process analysis and design for the development of multi-agent systems*, pages 186–197. Springer-Verlag, Berlin, Heidelberg, 2003.

[16] G. Leandro. *Meta-analysis in Medical Research,The handbook for the understanding and practice of meta-analysis*. BMJ Books, 2005.

[17] V. S. Massimo, M. Cossentino, and S. Gaglio. A repository of fragments for agent systems design. In *Proc. Of the Workshop on Objects and Agents (WOA06)*, pages 130–137, 2006.

[18] A. Mercier and M. Beigbeder. Calcul de pertinence basée sur la proximité pour la recherche d'informations. 9:43–60, 2006.

[19] R. L. Milidiu, T. Melcop, F. d. S. Liporace, and C. P. d. Lu. Simple - a multi-agent system for simultaneous and related auctions. In *IAT '03: Proceedings of the IEEE/WIC International Conference on Intelligent Agent Technology*, page 511, Washington, DC, USA, 2003. IEEE Computer Society.

[20] S. Russell and P. Norvig. *Artificial Intelligence: A modern approach*. Prentice Hall, 2009.

[21] V. Sugumaran and V. C. Storey. An ontology-based framework for generating and improving database design. *Natural Language Processing and Information Systems*, 2002.

[22] A. J. Sutton, D. R. Jones, K. R. Abrams, T. A. Sheldon, and F. Song. *Methods for Meta-analysis in Medical Research.* London: John Wiley, 2000.

[23] Z. Zhang and D. Fenstermacher. An introduction to mama (meta-analysis of microarray data) system. *27th Annual International Conference of the Engineering in Medicine and Biology Society*, 2005.