Query's Reply Generating from Sensors Regarding the Consumer's Constraints Application Using MAS

Oussama Legha, Stéphane Perrin

Query's Reply Generating from Sensors Captured Information Regarding the Consumer's Constraints

Oussama Legha
Polytech'Savoie - LISTIC
Savoie University
Annecy, France
oussama.legha@univ-savoie.fr

Stephane Perrin
Polytech'Savoie - LISTIC
Savoie University
Annecy, France
stephane.perrin@univ-savoie.fr

Abstract

We describe how to generate a reply to a consumer's query by taking in mind not only the needed information request but also the query constraints. The use of Directory Facilitator helps the consumer's interpreter locating an interface able to answer its demand and so, the supplier services that meet its needs. This interface by using a proxy cache is able, regarding the consumer constraints, to generate information replies to requests from earlier data acquisitions. Applying Multi-Agents System helps creating an effective consumer-supplier's interface with information treatments and data fusion processes. In this paper, we suppose architecture for the consumer structure and we show by an implementation how agents can handle and treat consumer requirements.

Keywords: Consumer-supplier, Intelligent Instruments Networking, Distributed Sensors, Fusion, Directory Facilitator, Multi-Agents System.

1. Introduction

In the world of intelligent instruments the supplier is the unit (instrument) that, by request or periodically, is able to generate and supply information data from sources like sensors. This information data could be values, measurements, etc. The consumer is the unit (instrument) that needs information to execute a task. A consumer could be an actuator or another supplier that needs complementary information for better task achievement.

1.1. Simple supplier and consumer exchange

The information exchanges between a consumer and a supplier can have different aspects depending on the method of conversation and the way the information is requested and its preparation to send for the consumer, and also on the representation of this information. To explain a scenario of this exchange, we present the following example:

The consumer's need of information can be presented by a simple query for requested information (Inf_{req}). The information request may also include preferences or constraints proposed by the consumer in way to better define its needs. This constraints could be the confidence interval tolerance *[]*, some related to time issues like the supposed time limit (deadline) to get the answer (t_{an}) when all responses exceeded this time are eliminated, some pre-selected answers elimination, etc.

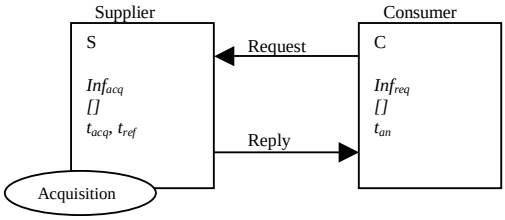

Fig.1. Consumer and Supplier structure

The supplier, on the other hand, holds a generated information (Inf_{acq}) from earlier data acquisition processes. It's possible to evaluate the information temporary confidence interval *[]* regarding its date of acquisition (t_{acq}) and, if known, the supplier can provide the next planned data acquisition date (t_{ref}) when (Inf_{acq}) will be refreshed. The supplier by taking part of the information treatment has to satisfy the

consumer request of information. It also has to handle the information received by acquisitions by saving them (on an internal memory or cache memory) considering their life time constraints. The confidence interval of this information may change in time depending on the information's variation in time [1]. By having the model of this variation and depending on the consumer request, the supplier is able to generate a reply from stored earlier collected data by evaluating the variations at the moment where the request is initiated. A scenario proposed by Perrin *et al* [2] describes this exchange's algorithm:

- The consumer transmits its request,
- The supplier computes the uncertainty from the information delivered by the last acquisition to give two ways to act:

1- Upon the demand, if necessary and possible, a new acquisition process will take place,

2- Otherwise, the supplier replies by the available information and gives the decision to the consumer: either to accept or to refuse or maybe to wait a newest acquisition.

To mention here that, by transmitting the information reply to the consumer and giving him the decision to select, an undesired traffic appears and the exchange between the supplier and the consumer takes more the form of a dialogue which is not allowed in some networks. To limit this traffic, the consumer can preview his decision within the request explaining how he would act: whether to take the available reply (for this example) or to wait for a fresh acquisition. The pre-decision could be defined by the supplier regarding the constraints that consumer applies into the request.

The example above presents the case for one supplier and one consumer while the existence of several suppliers and consumers could apply a real problem especially in an environment where the time related issues are important (real time networks). The supplier, with minimized limited capacity, could not be able to execute the requests of more than one consumer at the same time. Beside, the exchange communication between the consumers and the suppliers should be supervised and moderated to prevent collisions, loses and network's overload. That's why the idea to have an intermediate (interface) appeared.

1.2. Interface functionalities

To handle this communication, Perrin *et al.* [1] propose the use of interface that should handle the communication and the exchanges. This interface has to manage the requests and the replies by receiving the request of each consumer and providing the replies to requests. Also, the interface takes in charge a part of the computing process by evaluating the confidence interval for each acquisition date then it selects or mixes (in fusion process) the results to provide a reply that meets the consumer needs. The interface plays the turn of an indispensable mediator playing the matchmaker between the consumers' requests and the suppliers' generated information. The interface doesn't necessary present a real independent entity or an intermediate level between suppliers and consumers since it could be integrated in each of them or initiated (its functionalities) on a platform.

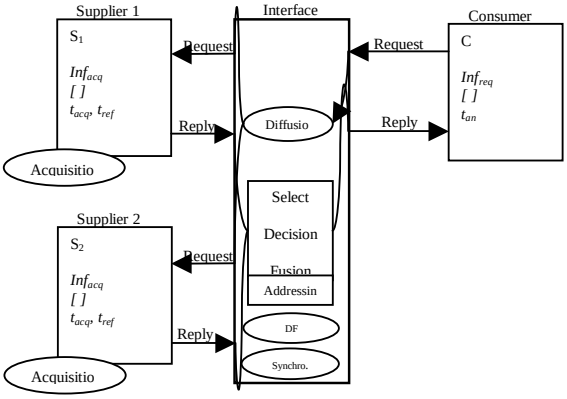

Fig.2. Suppliers, Interface and Consumer structure

The interface handles the communication issues (Directory, addressing, subscriptions, and time synchronization) and the connecting/disconnecting of instruments by managing and providing the connection sockets depending on the applied communication protocols. Beside, the interface has to handle the following issues:

- Managing the addressing for each instrument,
- Providing directory services,
- Managing the information (or data) propagation (diffusion) between instruments,
- Managing the connections streams and priorities (make possible the subscriptions between consumers and suppliers),
- Managing the information dating and the instruments time synchronizations.

The use of a Directory Facilitator (DF) makes possible the look-up of services in way to have the matchmaking between queries (requirements) and replies (satisfaction).

1.3. Directory Facilitator functionalities

In large definition the DF is a kind of "yellow pages" service adapted to the peculiarities of pervasive

computing environments. It's like a shop assistant. New entrants (consumer) to the shop will want assistance in order to better locate items of interest (suppliers). The DF has usually a centralized form integrated in the interface related to a platform usually specified as "main". The DF on a main platform is also called a central DF. After its setup, the DF requests from each instrument a self-identification by sending its ID information. These identifications are stored in directories. The identification may contain the following information:

- The type of instrument (supplier, consumer, etc.) "*InsT*",
- The instrument services specifications: service description "*Sd*", information type "*InfT*", information providing/consuming method "*Method*", response delay "*RespD*", information estimated life time, and confidence interval "*[]*".

The DF columns are predefined by titles. Identifications should be stored in order. By having this information, the DF is able to reply to interrogations concerning the connected instruments identifications and the services locations.

2. Multiple suppliers and Consumers exchanges

Connected to the system (platform), the suppliers and consumers response to the DF request by identifying themselves. These identifications stored in the DF help establishing and managing connections and later information exchanges between suppliers and consumers. After when the connections are established, the use of DF may become secondary, unless for updates when changes took place.

2.1. Information exchanges managed by DF

Once connected to the system, in way to resolve the consumer query, a communication scenario takes place like the follow (Fig.3):

1. Suppliers provides to the DF their specifications, DF verifies that suppliers are not already registered,
2. New Suppliers' specifications are stored to DF Data Base,
3. Consumer sends its information request presented by a query,
4. Processing Unit (PU) looks-up in the DF for suppliers that meet the consumer's demand,
5. DF matchmaking diverts the query to the chosen suppliers,
6. Chosen suppliers send their data replies to PU,
7. PU uses the received data to prepare the information for the consumer reply and returns it back.

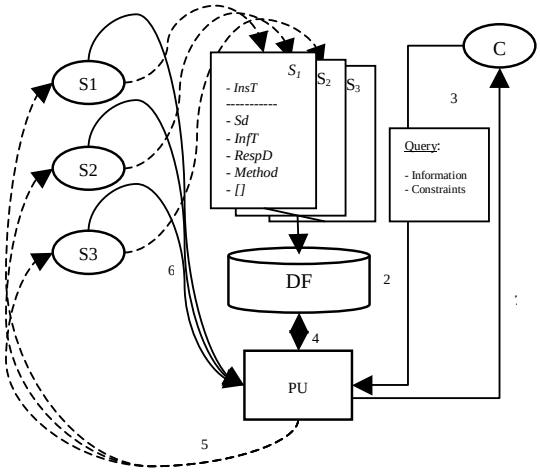

Fig.3. Exchanges managing using DF.

The DF could be integrated in an interface. It could also represent an interface by publishing its capacities. Beside the DF, interface could integrate other functionalities like data memorizing, information processing, etc.

When the supplier is programmed to function in push mode, the interface keeps a copy of the acquired data in its cache.

2.2. Instruments advanced interface

The interface processor is able to handle some information processing and also to manage the use of DF. The interface provides a similar function as a proxy server for consumers and suppliers. It intercepts all requests sent by the consumer to see if it can fulfill the requests itself. If not, it forwards the request (or a part of it) to suppliers by asking for new data or just information updates. The use of this method may improve the system performance by saving the results of acquisitions for a certain amount of time (till next update) so if a consumer requests information, the interface returns the latest information results it has generated using earlier acquired data or an earlier consumer's request results if these results meet the consumer's request's requirements.

We suppose that the consumer query (information request) is formed by two main parts:

1. The kind of requested information (measurements, image, value, etc.),
2. The requirement's constraints (time constraints, tolerances, precisions, error limitations, etc.).

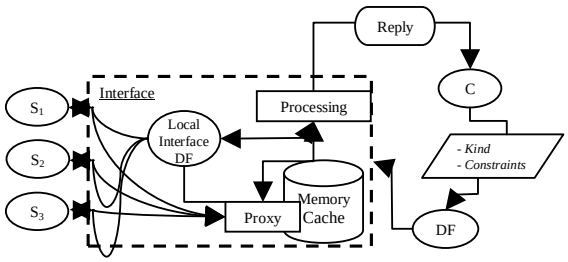

Fig.4. Advanced Interface architecture.

It's necessary that the interface (illustrated in Fig.4) knows the related suppliers. This function is guaranteed by the Local Interface's DF. On the other side, the consumer has to locate the interface(s) that supply the desired kind of information. The query of a consumer *C* interrogates a DF containing information about interfaces to locate the interface that it could connect to in way to satisfy its needs. After locating the interface, the DF manages the connection and updates its registry data base. The interface receives the consumer's query, determinates if it's possible to satisfy the query by evaluating measurement using data acquisitions information received from sensors and supposed known variation model. This variation model gives the estimation of measurement between two acquisitions. The time delay token by the data to reach the interface from the suppliers, compared to global system global time delay and capacity, is considered as negligible. The suppliers' provided data (measurement) could be synchronized and stamped by their arriving dates to the interface.
The interface regarding the consumer's constraints determinates an appropriate response to return. This process could be applied by simple select method or by more advanced mode by using knowledge variation's model (see section 3.4), as well by applying fusion process.

2.3. Conclusion

In section 2.1 we show how the DF could be used to manage the information exchange and by using the DF locating services capacity, the consumer gets in connection with the interface that meets the requested information's kind. In the same way, the interface gets connected to the suppliers. The interface with its suppliers has a data fusion capability and information proxy cache memory so it could process the consumer's query and return back a reply with condition that this reply is generated regarding the consumer requirements (constraints).

3. Information exchange using MAS

3.1. Introduction

There are a lot of definitions describing what an Agent is. In a generic sense, an agent is an entity capable of carrying out goals and it has two key properties: a partial autonomy and being a part of a community in which mutual influence occurs [5, 7]. In informatics' world, the main reason for the popularity of Multi-Agent Systems (MAS) is that modern computing and information environments are distributed, large, open and heterogeneous and Agents are able to act upon the environment in an autonomous behavior. Agents are also able to act in environments related to real-time issues [5].
In MAS environment each agent has incomplete information and capabilities. There is no global system control, data is decentralized and computation is asynchronous.
Using agents for information exchange between suppliers and consumers will guarantee the conditions for a distributed aspect of intelligent instruments. Agents are able to present every part of our structure. The suppliers could be presented by agents managing their functionalities and also the consumers. The consumer interpreter (presenting the consumer C in section 2.2) could be presented by an agent that could handle the information gathering and fusion. Presenting DF by an agent allows it to have a distributed aspect. This aspect is possible by cloning and sending the DF agent's copies on other connected platforms, by having part of its information copied on the working agents or by distributing DF fragments [8].

3.2. MAS structure

As we saw in section 3.1, MAS is composed of a number of agents that are able to interact with each other and with the environment. They don't have the same capabilities and they have different knowledge about the environment.
Consumers & Suppliers are defined by a reactive agents' model. They are composed of two modules: a knowledge module and a communication module. The knowledge module contains individual information about the agent and information about the calculation procedures.
Consumer interpreter into the interface is defined by a cognitive agent model. It's composed of four modules: a knowledge module, a strategy module, a communication module and a memory module. These

modules help the agent gathering and processing data to generate the consumer requested information.

MAS platform, before the agents (representing consumers and suppliers) are firstly initiated. A platform could be initiated with some predefined agents carrying the platform main functions (DF, connections, debug, etc.). An agent could represent one or more functions. In this part of the section we will detail the functionality of some agents.

- DF agent: kind of cognitive agent model provides directory's services and handles information about other agents.
- Synchronization agent: it date-stamps the arrived data (from suppliers) or information (requests and replies) to the interface. The date could be generated by the agent itself or just by using an external clock (PC clock, synchronizer clock, etc.).
- Communication agent: it manages the connection to the platform by alerting DF about new connections and by capturing the connected agents' status.

We don't deny the possible presence of other agents (may be used like administrator's tools on the platform: Debugger, Sniffer, etc.) we simply don't focus on them in this paper.

3.3. MAS implementation

In this section we present the implementation of the interface (presented in section 1.2) lately upgraded in section 2.2. The interface, by its functionalities, represented by agents, is integrated in a MAS platform.

We suppose that there are n intelligent instruments (consumers and suppliers) initiated in the platform. These intelligent instruments are presented by agents that handle their functionalities. We have S_i agents presenting the supplier, C_j agents presenting the consumers.

In the figure below (Fig.5) we illustrate the implementation structure. Every agent, setup method begins with the agent registration to its local DF by providing their predefined individual information as (for supplier): available S services, data acquisition refresh date T_{ref}, information confidence interval I_{conf} with variation admitted as known. The consumer consumer's agent registration to the DF is not obligatory. In some cases (with mobility applied), this registration could be needful to help managing the exchange (by registering some important information about the consumer as its location address). The DF saves this information in its directory. The DF list is built from services and agents' individual information that have been heard on the platform. We suppose that each line of this list has three fields: the service description Sd, the supplied information refresh date T_{ref} and the evaluated confidence interval I_{conf}.

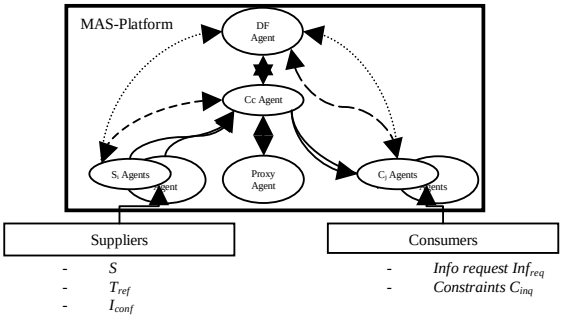

Fig.5. MAS Suppliers – Consumers structure.

The need of information request is presented by a consumer agent "Agent C" (or a set of agents Agent C_j). Its request of information is presented by a query (section 2.2). The consumer agent interrogates a local DF to look-up the interface that may fulfill its needs. When the interface is located, the query is diverted to its new destination. It causes the creation of an interpreter agent "agent Cc" (section 3.2) with a mission to serve the consumer's needs. This agent, interpreting the consumer to the interface (replaces the PU in section 2.1), interrogates the interface's local DF to project the following steps. By using the DF, the interpreter agent Cc locates the services it may need on the interface and takes in duty the processing tasks. It contacts the proxy agent by requesting available data then; it verifies the data validations and tries to compute the variations done by the time. The scenario of section 2.2 describes the steps Cc has to follow: depending on the consumer request constraints agent Cc decides whether to use the available data or to locate suppliers that could provide newest acquisitions in way to prepare the request's reply by negotiating the task resolving with them. When the reply is ready, agent Cc sends it back to the consumer. A copy of the results is always stored by the proxy agent for later uses. Agent Cc, after reaching its goal, changes the status from active to inactive to be terminated later.

3.4. Application

In this paper, for our demonstration, we use JADE (Java Agent Development framework) [9] with its platform and predefined main functions as our Multi-Agents System platform. The decision to use JADE is justified by the advanced MAS platform it offers with advanced and useful predefined functions we may need.

We initiate the JADE platform with its main container and default agents. Then, two agents (S_1, S_2) representing the suppliers are initiated. These agents provide information about measurements captured by sensors. These sensors observe the same physical phenomena to get the physical value of the measurement "M_{inf} ". They are characterized with different parameters concerning their acquisition time, refreshment's frequency, and precision. For this application, we admit that we can evaluate the physical environment variation limits so we can evaluate the confidence variations by time. The variation for each supplier is modelized by maximum and minimum possible variations. These are considered bounded and linear; we suppose that the slopes are known (see Fig.6).

For suppliers we predefine the following specifications:

- Its provided services description $S_d(i)$,
- The measurement's information $M_{inf}(i)$,
- Its data acquisition method $AM(i)$,
- The refresh time period $t_{ref(i)}$,
- The in time evaluated confidence interval $[CI(t)]_{(i)}$.

The suppliers' information is presented like the follow:

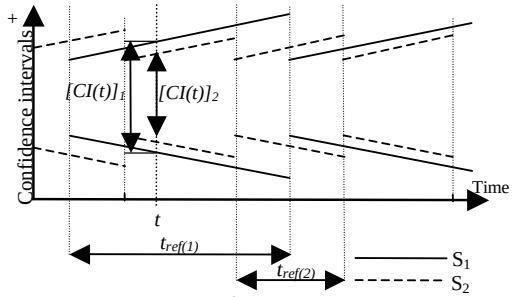

Fig.6. Suppliers' confidence intervals variations.

At the same calculation time t_c, suppliers' confidence intervals (presented by segments) are not the same.

$$[CI(t_c)]_1 \neq [CI(t_c)]_2 \qquad (1)$$

For the consumer, on the other hand, we specified an agent named agent *C* which has to detect the client query. To our system, the agent *C* presents the consumer.

Connected to the platform and after registering to the local DF, agent *C* uses the DF services to locate an interpreter that can take in charge the query. After locating the interpreter (Agent *Cc*), agent *C* transmits its query. As we described in section 2.2, this query is composed of two main entries (requested information kind and constraints):

1. The measurement's information he is looking for M_{req},
2. Constraints:

- The allowed delay limit to have the answer T_{resp},
- The predefined confidence tolerance $[CT]_{req}$.

The measurement's information is predefined for further works when there are different measurements types on different suppliers. The illustration of the variation model is showed in the figure below (Fig.7).

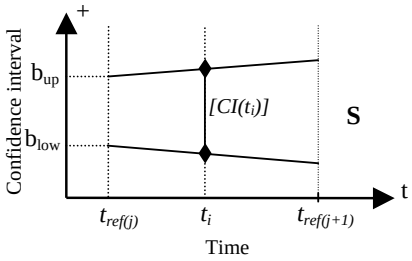

Fig.7. Variations' model.

For a time moment t_i we can evaluate the confidence interval segment for information obtained by supplier S:

$$For\ \ t_{ref(j)} \le t_i \le t_{ref(j+1)}$$
$$[CI(t_i)] = [y_{up}(t_i); y_{low}(t_i)] \qquad (2)$$
$$with: \begin{cases} y_{up} = a_{up} \times (t_i - t_{ref(j)}) + b_{up} \\ y_{low} = a_{low} \times (t_i - t_{ref(j)} + b_{low} \end{cases}$$

Where: a_{up} and a_{low} are the maximum and the minimum variation speeds. [b_{up} ; b_{low}] the confidence interval of the measurement (updated at the moment t_{ref}).

In our application we have two suppliers so the interface has to return the evaluated confidence interval in the moment of request, and to find out whether this evaluation is conformed to the requirement's constraints or not.

The consumer in its constraints defines the request response delay (T_{resp}) that gives a set of confidence intervals for every time moment.

In this application case, we suppose that the confidence interval tolerance $[CT]_{req}$ is the minimum possible value evaluated during T_{resp}.

For our case study (illustrated in *Fig.8*) we suppose we have two requests, at the same moment, for the same information but with different contexts: the response delay time T_{resp} is different.

If the available evaluated information is suitable to the request tolerance the interface return back the computed value.

Else, if an update takes place during the request's response delay and if the sensor precision is suitable to

the request tolerance the interface has to decide to wait the fresh value of the acquisition.
In other case, the interface could give the computed value or an empty answer depending on the programmer choice.

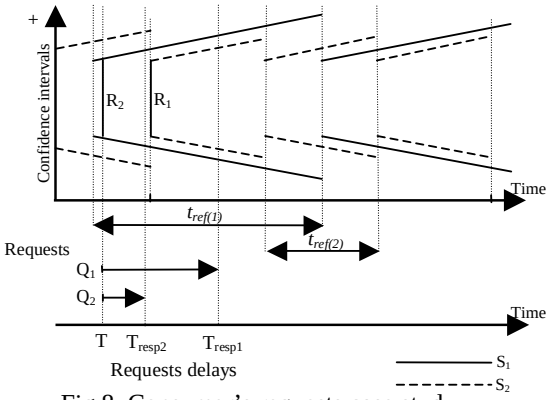

Fig.8. Consumer's requests case study.

With Q_1 and Q_2 present respectively the first and second query on the time scale. R_1 and R_2 present respectively the reply to the queries Q_1 and Q_2. T_{resp1} and T_{resp2} are the response delays limit.
Using several suppliers characterized by different specifications (especially precision) the case could became obviously complicated.

3.5. Application's numerical results

In our academic case, we defined the physical phenomena's variation by real equations like the follow:

$$\begin{cases} [IC(t_{refi})] = [3.0\,;\,6.0] \\ a_{up} = 0.25\,unit/ms. \\ a_{low} = -0.25\,unit/ms. \end{cases} \qquad (3)$$

And the time parameters:
The date difference between S_1 and S_2 is T_{diff}= 350 ms.
The refresh period of S_1 is t_{ref1} = 140 ms.
The refresh period of S_2 is t_{re2} = 70 ms.
The first request delay is T_{resp1} = 90 ms.
The second request delay is T_{resp2} = 30 ms.
Requests are initialized at the date T (Section 2.2).
Fig.8 illustrates the time diagram for our case study.

The answer (M_{inf} section 3.4) to the first request is:

M_{inf} with *[CI(t)]* = **[3.0 ; 6.0]** at T_{req1} = T + 290 ms

The second request returns the following answer:

M_{inf} with *[CI(t)]* = **[2.75 ; 6.75]** at T_{req2} = T + 0 ms

We note that the two answers are different even when the corresponding requests are initiated at the same date and they are related to the same information kind. The results difference is caused by the different consumer's needs.

4. Conclusion and perspectives

A consumer with the ability to express its context (request constraints) could obtain an answer that meets its needs. This approach will improve the information treatments especially the decision process. Information processing to generate the answer adapts itself to the consumer constraints (context).
The proposed approach allows the system, by regarding the consumer's constraints, to determinate the best answer. The definition of "best" is expressed by the consumer. The answer could be improved by using fusion process.
We illustrated our approach by an implementation using MAS. The use of MAS is motivated by its capacity of holding a mobile aspect. Mobile agents are often used for distributed information retrieval and information dissemination [3, 4]. Transmitting the computation engine instead of the data may improve the repartition of data treatments and makes possible the distributed fusion aspect [7].

References

[1] S. Perrin, E. Benoit, L. Foulloy, *Temporal Information Modelling in the Context of Intellignet Instruments*, In: Proc. Of 5th IFAC Int. Symp. On Intelligent Components and Instruments for Applications, pp. 257-263, Averio, Portugal July 2003.

[2] S. Perrin, E. Benoit, L. Foulloy, *Integrated information consumer need based on information and accuracy modeling with time consideration*, 10th IMEKO TC7 International Symposium, Saint-Petersburg, Russia, July 2004.

[3] C.C. Hayes, *Agent in a nutshell*, Introduction IEEE Transaction on Knowledge and Data Engineering, 11(1), pp. 127-132, 1999.

[4] J.S. Wong, A.R. Mikler, *Intelligent mobile agents in large distributed autonomous cooperative systems*, Journal of Systems and Software, 47(2), pp. 75-87, 1999.

[5] G. Weiss, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, MIT Press, Cambridge, ISBN 0-262-13103-0, pp 37-51, 1999.

[6] T. Oates, M.V.N. Prasad, V.R. Lesser, *Cooperative information-gathering: a distributed problem solving approach*, IEEE Proceeding – Software Engineering, 144(1), pp. 72-88, 1997.

[7] H. Qi, X. Wang, S.S. Iyengar, K. Chakrabarty, *Multisensor data fusion in distributed sensor networks using mobile agents*, Proceedings of Information fusion, 2001.

[8] C. Campo, *Directory Facilitator and Service Discovery Agent*, Universidad Carlos III de Madrid, July 2002.

[9] Java Agent Development framework: http://jade.tilab.com.