Computers in Industry 91 (2017) 11–23

Contents lists available at ScienceDirect

Computers in Industry

journal homepage: www.elsevier.com/locate/compind

Integration in industrial automation based on multi-agent systems using cultural algorithms for optimizing the coordination mechanisms

Juan Terán[a], Jose Aguilar[a,c,*], Mariela Cerrada[b,d]

[a] CEMISID, Dpto. de Computación., Universidad de Los Andes, Mérida, Venezuela
[b] CEMISID, Universidad de Los Andes, Mérida, Venezuela
[c] Prometeo Program researcher, Escuela Politécnica Nacional, Quito-Ecuador, Universidad Técnica Particular de Loja, Ecuador
[d] Prometeo Program researcher at GIDTEC, Carrera de Ingeniería Mecánica, Universidad Politécnica Salesiana, Cuenca, Ecuador

ARTICLE INFO

Article history:
Received 26 August 2016
Received in revised form 15 December 2016
Accepted 23 May 2017
Available online xxx

Keywords:
Integration
Industrial automation
Multi-agent systems
Cultural algorithms

ABSTRACT

Integration in industrial automation can be approached from the theory of Distributed Artificial Intelligence. One approach is the modeling of different production units by agents that interact through interaction protocols, which are implemented following a coordination mechanism. Under this approach, integration in automation can be achieved through the optimization of implicit interactions in such mechanisms. This paper presents a strategy for integrating industrial processes based on Multi-Agent Systems (MAS), which consists of optimizing coordination mechanisms that implement conversations between agents, by using cultural algorithms. The cultural algorithm uses formal models of interaction protocols between agents, such as auction and tender, and the integration scheme comes from automation architectures based on MAS, to which their interactions are optimized. The proposed scheme enables data and service-oriented integration. The proposed strategy is applied in two industrial case studies related to the oil production process.

1. Introduction

The trend of automated systems is toward distributed architectures, whose heterogeneous components of software and hardware interact among them, seeking to achieve common goals that optimize the process performance. In this context, the integration of different existing technologies in industrial environments has become a key factor in the production process, and particularly, in the automation strategies of these processes [1]. Fundamentally, the integration is based on the development of models for the information exchange, the cooperation and the common data representation, to coordinate and implement actions or objectives. A definition of integration proposed in [1] is "the coordination of the operations of all company elements working together, to achieve an optimal compliance of the company mission".

There are two major approaches to system integration architectures: data-oriented integration (DOI) and service-oriented integration (SOI). DOI involves combining data that reside at different sources and platforms. DOI provides the users a unified view of such data [2]. SOI flexibly handles the problems associated with heterogeneous and legacy systems, enabling the organizations to offer existing applications as reusable services [3].

SOI is defined in [4] as the integration of computational entities using services based on service-oriented architecture (SOA). SOA is a software architecture based on software components, which provide functionality to applications as services. A service is an autonomous logical representation of a repeatable function or activity [4]. Several works address the combination of the paradigms of agents and services. In [5] the services are used to provide a generic scheme of interaction between agents. In [6], an extension of Web Services (WS) is proposed through an integration tool called WS2JADE, to manage the WS that are framed by the FIPA standards for MAS. One approach for integration in industrial automation is the definition of architectures based on service-oriented MAS.

Other works state that MAS is one of the approaches for orchestrating of integrated automation activities, through distributed architectures that implement the required services at different levels of automation [5–13]. Having as reference agent-based automation architectures, such as those ones in the works presented previously, the integration can be accomplished in a

* Corresponding author at: CEMISID, Dpto. de Computación., Universidad de Los Andes, Mérida, Venezuela.
E-mail addresses: carlostp@ula.ve (J. Terán), aguilar@ula.ve (J. Aguilar), cerradam@gmail.com (M. Cerrada).

http://dx.doi.org/10.1016/j.compind.2017.05.002
0166-3615/

vertical or horizontal manner. Vertical integration is the flow of decisions/actions between components that are in different levels of the architecture, and the feedback derived from the same flow (inter-level flow). Horizontal integration arises due to the flow of products and information in the same architecture level (intra-level flow). In vertical integration, the decision management occurs in a hierarchical architecture where the different functions of the company are associated with a hierarchy. In horizontal integration, different components in the same level perform complementary activities, without requiring information units of other level [1].

Through a suitable multi-agent modeling, which implements the basic functionalities of an industrial automation system as services, it is possible to achieve the expected integration through the coordination of agent interactions.

Recently, a system based on cultural algorithms (CA) [14] has been developed to address the coordination of MAS from the perspective of the collective learning approach [15]. The collective or distributed learning is carried out by groups of agents (e.g., by sharing knowledge, or by observing other agents) [15]. One of the most common coordination mechanism in the literature is the negotiation, and its interaction protocols are auction and tender [16]. Their representations in formal mathematical models are presented in [17], where each conversation between groups of agents is implemented through the combination of protocols properly selected and instantiated by a CA. MAS's agents learn to discern the most convenient coordination mechanism for a conversation, through a cultural model. In [18], the learning model based on CA is called model of cultural learning, and it aims to optimize interaction protocols, in which the speech acts are performed in order to achieve the coordination of interactions between agents.

In this paper, we use the formalism presented in [15,17,18], to optimize the coordination mechanisms of the conversations in MAS that are modelling industrial automation systems, as an integrating strategy of automated environment. This strategy allows the integration of the services provided by agents and the resource management for the development of activities, which includes the utilization of data shared between all agents. Thus, by optimizing the coordination mechanisms in the conversations of the different agents at different levels of abstraction, a better integration between them is guaranteed for the development of industrial activities. In this sense, the services of industrial automation offered by the agents are executed during the conversations, with proper interaction protocols which are selected by an optimization process using CA. This allows performing different forms of integration (horizontal and vertical, oriented to data and services), based on the coordination of interactions between agents.

Our work is based on known ideas about MAS modeling for automation, which are properly used to construct our contribution. The interactions between agents, i.e. conversations, to accomplish the activities and tasks in the corresponding automation process, are described through coordination and communication models, and the set of conversations determines the coordination scheme of the MAS. In our knowledge, there are not researches widely reported that propose the optimization of MAS coordination from the perspective of collective learning, where processing and communication costs are the main aspects to consider for real time implementations, such as the applications in industrial automation. We only need the specification of the interaction type characterizing in a general manner the communicative acts between agents, how many agents are assumed to implement the service, and the variables describing the processing and communication costs. Consequently, our approach avoids the necessity of having empirical data for validating our optimization approach.

Particularly, our optimization model is based on an evolutionary algorithm working off-line, which evaluates different interaction protocols among the agents, in each algorithm's iteration. One conversation can have several interaction protocols, for each communicative act between agents in such conversation. As a consequence, in each algorithm's iteration, our approach proposes and evaluates different coordination schemes for the same MAS, and the most convenient scheme regarding some objective function is considered as a belief by all the agents in the MAS.

The main contributions of our paper are summarized as follows: (i) The generic modeling of the interaction protocols proposed by FIPA, such as auctions and tender, (ii) The modeling of vertical and horizontal integration in industrial automation through a MAS architecture, using proper conversations as coordination mechanism, (iii) The generic characterization of conversations in any MAS by defining types of interaction, such as: consult, assign, inform, and request, (iv) The development of a collective learning approach based on a cultural algorithm, which evaluates the best set of interaction protocols for implementing the generic types of interaction in the conversations of a MAS, as a way to reach data and service oriented integration in industrial automation tasks.

This paper is organized as follows. Section 2 presents a review of recent results about the use of MAS oriented to services and their coordination, applied to industrial automation. Section 3 describes the optimization model using CA for coordination of the conversations in MAS, which requires the characterization of the interaction types (IT), and uses formal models of the interaction protocols, such as auction and tender. Section 4 proposes the automation integration approach based on coordination of MAS, and several general premises are presented for the integration under automation architectures using MAS. Section 5 shows two case studies to test the proposed integration approach, and discusses a comparative table between our approach and integration models proposed in other works. Finally, Section 6 presents the conclusions.

2. MAS, services and coordination in automation architectures for industrial applications

The use of MAS and WS in industrial automation applications have been reported in the literature, where service-oriented architectures supported by MAS have been defined. The work in [7] points out that the combination of WS and MAS provides a promising computing paradigm for the selection of an efficient service, and the integration of business processes between organizations. In that paper, the goal of WS is to support business collaboration to achieve high-level of distributed and reconfigurable integration. In [8], the production process is seen as a collection of services. The coordination of services, handled by an orchestrator, and the execution of the service sequence, handled by an orchestration engine, is offered by a service-oriented middleware; this middleware also includes a decision support system, which is integrated through MAS for conflict resolution between the orchestrator and the orchestration engine. Finally, to integrate existing legacy MAS applications, messages between agents described by the Agent Communication Language (ACL) are translated to a service-oriented message (WS-ACL) that can be treated by the mentioned middleware. A similar approach, where agents are used for the composition of services by negotiation, is presented in [9]. The integration architecture is a hierarchical structure where the lowest level is the level of services. Each service is modeled as an agent, and the agents negotiate the

composition according to their service objectives, in the negotiation level.

Another approach regarding the integration in industrial automation, is the definition of architectures based on service-oriented MAS. The work presented in [10] proposes a set of agents requiring or providing resources as services. In particular, it describes a model for the development of MAS, which offers and/or requires services to meet their goals in a specific domain, in this case, the domain of production and industrial automation. In general, resources, agent functions and other software components are seen as services. A service-oriented architecture of MAS, as described in [10], is presented in [11]. The architecture is defined by different nested abstraction levels, each one modeled by a MAS. In the outermost level the components of the production process are modeled, also called business objects; automation activities are modeled in the mid-level, such as control, supervision, production planning, managing production factors, among others. In the innermost level, the functionalities of the medium level are modeled through generic agents with the capabilities of observation, execution, control and coordination, seen as agents of an automatic control loop [12]. Each agent of this architecture is specified by using the methodology called MASINA (Multi Agent System for Industrial Automation) [13,19]. This proposal can implement service-oriented integrated automation applications, with the flow of information and requirements from inter- and intra-levels. The heart of MASINA relies in the definition of services to be provided by agents, with conversations and their speech acts, which describe the dynamics of MAS through the consumption of services.

The agents in a MAS interact with each other using protocols and high-level communication languages standardized by FIPA [20,21], to solve problems that are beyond the capabilities or knowledge of each one [16]. The interaction between agents is achieved through coordination schemes, which can be described as the set of supplementary interactions (speech acts) needed by an agent community, to act collectively in an integrated manner. Sets of interactions are grouped into conversations, in order to accomplish the objectives of the MAS. Coordination in MAS is not limited to sending messages between agents (speech acts), but in the manner as an agent interacts with other agents to provide or require services. There are different reasons why agents need to be coordinated, such as to prevent chaos and anarchy, overcome local limitations, share resources and information, among others [22].

There are several techniques to achieve the coordination of the different agent communities. For example, in [23] multiple coordination corresponding to a diversified synchronous rate is studied, where different collective behaviors can emerge into several sub-groups of a MAS. The idea is the integration of each subgroup through agreements with each other, by performing convergence analysis using distributed control algorithms based on techniques of graphs and matrices. In [24] the coordination based on patterns for automated production systems is studied. This architecture consists of three main parts: a simulator, a schedule system of work orders, and a management system of execution tests. The coordination patterns in the MAS describe communication and negotiation between groups of agents, specifically between the agents of the simulator and the schedule system, for the allocation of available resources.

An approach from artificial intelligence to coordinate MAS in industrial automation applications can be reviewed in [25], it proposes a module with rule-based reasoning capability that translates work requirements of customers in a set of basic tasks that must execute a group of virtual agents. In [26], the issue of MAS-oriented software engineering is addressed, and how it has succeeded in the design of organizational structures, knowledge representation to coordination problems, among others. The paper proposes an architecture of agent-based intelligent integration. As a case of study, it analyzes the integration requirements for information systems of the electric industry. In [27], the selection of coordination mechanisms in agent's societies is studied, to allow migration of users and services between MAS. The idea is to propose a structure to help choosing a coordinating mechanism for the integration of MAS in an intelligent ecosystem. The work presented in [28], investigates the problem of hybrid coordination in multi-agent networks with hierarchical leaders. First, in the case of static main leaders with groups of followers uncoupled, a necessary and sufficient condition is given to all leaders to obtain an expected formation, and so, followers groups converge asymptotically to their desired states. Second, in the case of static main leaders, but with coupled groups of followers, here they are provided with a unique condition on the control parameters and an adjacency matrix with weights, for the hybrid coordination. Third, in the case of movable main leaders with hierarchical delays, the condition provided to the multi-agent network allows deriving a desired group behavior.

The use of collective intelligence for the coordination of MAS in industrial applications has been reported in [29,30], which presents a scheme of coordination and control for MAS based on stigmergy for manufacturing systems. Stigmergy is a mechanism of indirect communication between individuals, through local modifications of their environment. In particular, the proposed mechanism is based on a technique used by the coordination system of insect colonies. An insect need only observe their local environment to consider nonlocal problems in their decisions. In this case, PROSA is selected as the MAS-based architecture for modeling manufacturing System [31]. In [32], the authors propose a scheme of intelligent control of multiple robots modeled as MAS from a multi-objective coordination model, which obtains formations based on the integration of intelligent control from the point of view of individual intelligence and global perspective of collective intelligence.

A wide review about managing production systems and guidelines for the design of complex systems in the framework of Holonic Manufacturing Systems is presented recently in [33], where the application of biologically inspired control paradigms to control complex manufacturing systems is discussed.

3. Formal model using CA for coordination of MAS

This section presents the model based on cultural learning for the coordination of MAS proposed in this paper. The model uses the mathematical formalization of the interaction protocols such as auction and tender, as well as the characterization of generic interactions called 'interaction types'. Section 3.1 presents these mathematical descriptions and Section 3.2 presents the proposed CA.

3.1. Formal model of interaction protocols

3.1.1. Interaction types in MAS

Exchanges of messages between agents exist in all MAS, which are needed to interact, coordinate, cooperate, and negotiate, among others. According to FIPA, each message sent or received by an agent is seen as a communicative act, and there is an extensive library of such acts [21]. When the sending of two or more communicative acts between two or more agents occurs, it is said that there is a conversation [34]. Conversations between agents generally have typical patterns that differ from each other. These patterns are determined by sequences that the communicative acts expected to follow. FIPA sets these exchange patterns as specific interaction protocols [34].

Each interaction protocol has a set of attributes, these are: roles of the agents, purpose of the conversation and a certain sequence of speech acts. Based on these attributes, there are four interaction types (IT) in a MAS, which frame a major part of the interactions reported by FIPA, which in turn allow the generalization of the conversations that can occur at any MAS [17]. These are:

- IT1: Consult. An agent looks for any information that can be stored in a database, repositories, warehouse, and Internet.
- IT2: Assign. Through this interaction, an agent orders the execution of tasks to other agents.
- IT3: Inform. An agent can inform to other agents about the occurrence of a certain event, or just processed information.
- IT4: Request. An agent demands a service.

Fig. 1 illustrates a conversation, in which the proposed IT are shown. For example, in the first interaction, the Agent One requests to the Agent Two the performance evaluation of a given activity (IT4). Subsequently, the Agent Two asks for locate the data needed to perform the task, to the Agent Three (IT1). In the next interaction, the Agent Two assigns a task to the Agent Four (IT2), and finally, Agent One reports the results (IT3).

Then, a conversation in any MAS can be characterized by these four IT, regardless tasks and objectives to be achieved.

3.1.2. Formal models of the auction and tender protocols

The auction protocol has been used in many practical applications of computer science that involve the allocation of goods, tasks and resources [17].

A group of agents, where one agent has the role of auctioneer and the remaining agents are bidders, makes up an auction mechanism. The classic scenario assumes that the auctioneer wants to sell an item at the highest price possible, while the bidders want to buy at the lowest price possible. These scenarios are developed through rounds, time or some threshold value. The auction starts with an initial price and ends when one of the bidder's proposals satisfies the Auctioneer. The proposal is declared as the winner, and takes the auctioned item.

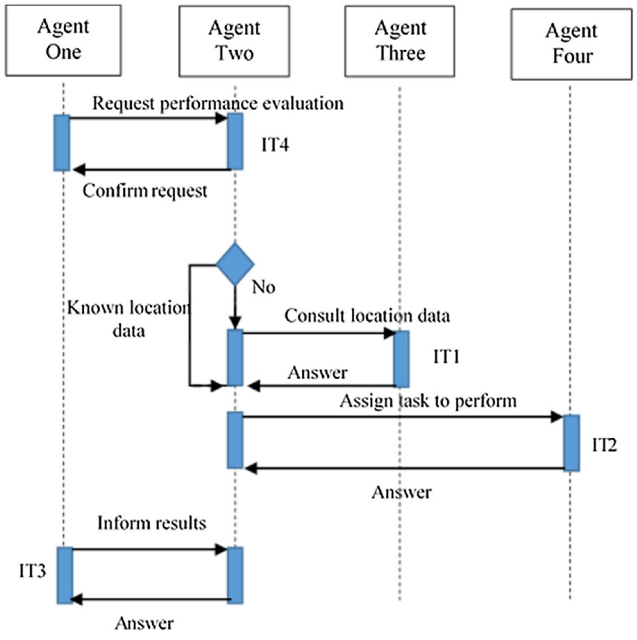

Fig. 1. Interaction types in a MAS's conversation.

Mathematical formalization seeks the parameters of the entire auction process between agents, and the auction is defined by the 6-tuple in (1). The complete formalization can be found in [15,17].

$$S = (C_0, Of_i^j, \vec{\varepsilon}_i, \alpha_i^j, C_p, C \quad (1)$$

where:

- C_0 is the initial price of the auction, $C_0 \in \mathbb{R}^+$
- Of_i^j is an offer matrix, $\in \Re^{(n+1)\times m}$ formed by the number of agents $i = \{1, \ldots, n\}$ plus a row vector A_G which shows the winner of each round, and the number of rounds $j = \{1, \ldots, m\}$.
- $\vec{\varepsilon}_i$ is a vector, where each entry represents the maximum amount of items that each bidder agent i can bid, $\vec{\varepsilon}_i = \{\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n\}$.
- α_i^j specifies a given proposal, being α the value proposed in the round j, and i the bidder agent.
- C_p is the condition for stopping the auction and it may be of different types:
- t (Time), if the auction is governed by time
- j (Number of rounds), if it is governed by rounds
- $\alpha_i^j = 0 \forall i = 1, \ldots, n$when there are not more bids from a given round
- x_r (Threshold), when a minimum price threshold auctioneer is reached (in the case of Dutch auction)
- C is the final price of the resource or item. $C = A_G^j$, $C \in \mathbb{R}^+$. It is the maximum value of the row vector A_G (for the case of the Dutch auction it is the minimum) and the winner A_G may be any element of the set A_i, i. e., $\exists A_i = A_G$.

In case of tender, is a system whereby adjudges something (an object, performing a work or a service) to the agent that offers the best conditions. Tender consists of a group of agents that can be managers or tenderers. When an agent has the role of the manager, it must perform the following activities:

- Break a complex task into less complex subtasks
- Announce to other agents that there is a sub-task expecting to be executed. These tasks are distributed through a broadcast message to all agents or they are addressed to a specific group of agents,
- Select the most appropriate offer when it receives the response to their requests,
- Assign a sub-task to winner agent, for which a contract is created,
- Monitor the progress of the contract, possibly asking for information, reports, among others. It is free to reassign the subtask if the contractor fails to complete, finally
- Integrate the partial results produced by the contractors in a complete solution.

Meanwhile, the tenderer agents perform the following activities:

- Receive announcements of tasks and assess their skills and availability to perform them
- Make an offer if they are able to satisfactorily perform the task
- Reserve the resources required for its execution, if an offer is accepted.

When the tenderers have won a bid, they must perform the assigned tasks, and generate reports about the progress of these tasks and the final results. Tenderer agents can become managers, if the subtask is too complex for their abilities, then they split the sub-task, and the assigning process of task is repeated.

The mathematical formalization of the bidding process is defined by the 8-tuple in (2), and the details are developed in [18].

$$L = \left(M, f(T), \vec{O}_c, g(O_c), M_p, RP, h_c(RP), RF\right) \quad (2)$$

where:

- M is the initial message, $M = (T, I_a, F)$ where T is the type of task expecting to be tendered I_a is the information of the manager agent (it can include its virtual address (DIR_a), etc.), and F is the expiration date for offering (it is given by a numeric pair $F = [DD, MM]$, where $DD \in \{1, 31\}$ and $MM \in \{1, 12\}$ are in time units).
- $f(T)$ is a function that allows assessing the capacities of the potential contractors for responding to the request for the execution of the task T.
- O_c is a vector, $O_c \in M_{m\times 1}$ with components O_i where $O_i \in O_{c,i=\{1,\ldots,m\}}$ contains the offer of each contractor i to complete the task T.
- $g(O_c)$is a function that represents the offer evaluation criteria of the manager agent.
- M_g is the message to the winner contractor. It is a 2-tuple $M_g = A_g, [accept - proposal]$, where A_g is the name of the winner agent, with the communicative act "accept-proposal". For the rest of agents, the manager agent diffuses a message that they did not win the bid.
- RP is a vector, $RP \in M_{1\times r}$ where each cell $rp_k \in RP$ represents the progress report k of the agent c that has won the sub-taskT_j (for $k = \{1, \ldots, r\}$).
- $h_c(RP, k)$ is an evaluation function of the manager agent to know the degree of task execution T_j by the contractor.
- RF represents the final report of the winner agent of the task T_j, and it is the last element of the vector RP of the winner agent, i. e., $RF \in RP$.

3.2. Cultural algorithm based learning model

The cultural learning model oriented to the optimization of conversations in MAS, uses formalisms presented in Section 3.1. The model is framed within the CA components [14]: a population, an objective function, a belief space and a communication protocol, as illustrated in Fig. 2.

In the following, these components are briefly presented, and how they were instantiated for coordination in MAS. The model can be viewed in detail in [15,17].

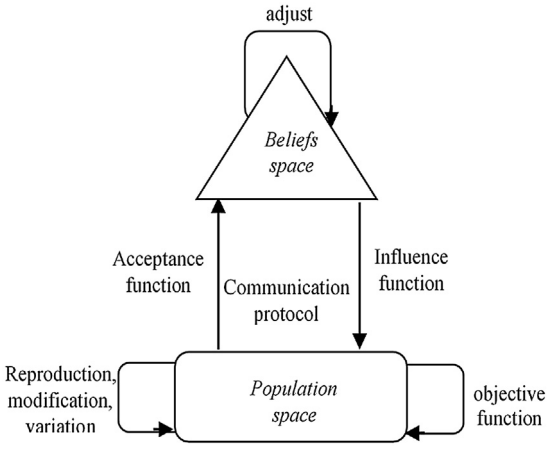

Fig. 2. Cultural algorithm.

3.2.1. Population

Each individual in the CA is a MAS composed by the different conversations performed by the community of agents. Every conversation, in turn, can have sub-conversations, which are characterized by the interaction types (IT), as defined in Section 3.1.1. Thus, the individual contains the set of conversations Ci, and a numeric value OF associated to the objective function. These individuals can evolve to form new individuals, through the genetic crossover and mutation operators. Fig. 3 shows the described individual.

C_i denotes an existing conversation i in MAS, OF is the individual value of the objective function, $C_{i,k}$ denotes the sub-conversation k of the conversation i, being m_i the number of sub-conversations associated with this conversation, IT is the interaction type associated to the such sub-conversations, $CM_{i,k}$ is the interaction protocol, CP_{ik} and CC_{ik} are the cost of processing and cost of communication while executing the interaction protocol, finally, $P^u_{i,k}$ are the u parameters for such protocol.

3.2.2. Objective function

This function allows evaluating the performance of each individual. The function is based on the processing cost and the communication cost, that are generated to run each communication mechanism used by the individual. The best individual will be those one that minimizes this function, which is defined by Eq. (3):

$$FO = \sum_{i=1}^{n}\sum_{k=1}^{m}(a * CP_{i,k} + b * CC_{i,k}) \quad (3)$$

where a and b are constants defined by the user, which allows normalizing the units in the function. The number of conversations is n, m_i is the number of sub-conversations in a conversation C_i. $CP_{i,k}$, and $CC_{i,k}$ are detailed in [15,17].

Cost of processing $CP_{i,k}$ is presented in Eq. (4):

$$CP_{i,k} = PI_k + PE_k + \sum_{l=1}^{j}\sum_{q=1}^{n_j} A_{l,q} \quad (4)$$

where PI_k is the initial processing cost of the mechanism, PE_k the cost of the execution process, $A_{l,q}$ is the allocation process that means the time it takes to prepare the proposals, n_j is the number of participants in the round j.

Cost of communication $CC_{i,k}$ is presented in Eq. (5):

$$CC_{i,k} = \sum_{l=1}^{j}\left(\sum_{r=1}^{N-1} CEP_{l,r} + \sum_{s=1}^{n_j} CEO_{l,s}\right) + \sum_{r=1}^{N-1} CS_r \quad (5)$$

where $CEP_{l,r}$ is the cost to send the initial proposal (initial cost), $CEO_{l,s}$ is the sending cost of offers, and CS_r is the cost to report who won. As mentioned, j is the number of rounds, N-1 the number of agents least who sends the message, and n_j is the number of participants in the round j.

3.2.3. Belief space

There are two categories of knowledge in the belief space: situational and normative.

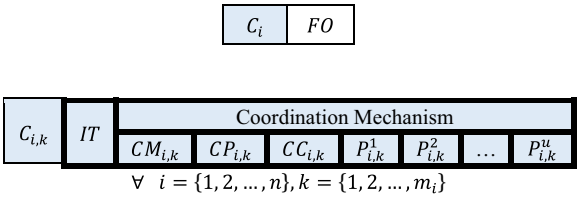

Fig. 3. Internal structure of the individual.

3.2.3.1. Situational knowledge. This knowledge contains examples of successes and failures of individuals. In the model, it is represented as the number of times that a particular IT uses an interaction protocol. Fig. 4 shows the structure of this knowledge, where *IT* is the interaction type, *CM* is the interaction protocol, *IO* is the index of occurrences of that CM for that IT at time t-1 (previous iteration), y *TO* is the total of occurrences of the interaction type IT, more details about this structure are given in [15].

3.2.3.2. Normative knowledge. It contains the desirable value ranges of each one of the parameters associated to the interaction protocol stored in the situational knowledge. In Fig. 5, the structure of this knowledge is represented, where *P* is the parameter of the interaction protocol, *UL* is the upper limit and *LL* the lower. For more details see [15].

3.2.4. Communication protocol of the CA

It is guided by the acceptance and influence functions. These functions allow the interaction between the belief space and the population, as illustrated in Fig. 2. The *acceptance function* takes 20% of the population, to feed back the belief space with their experiences.

The acceptance function allows updating both situational and normative knowledge in the belief space as follows:

For situational knowledge. Eq. (6) states the acceptance function:

$$IO_{(IT,CM,t)} = IO_{(IT,CM,t-1)} + \frac{NO_{(IT,CM,t)}}{TO_{(IT,t)}} \tag{6}$$

where $IO_{(IT,CM,t)}$ is the new index of occurrence at time t for an interaction type *IT*, and an interaction protocol (*CM*); $IO_{(IT,CM,t-1)}$ is the previous index of occurrence (t-1), i.e., which is currently in the belief space; $TO_{(TI)}$ is the total number of occurrences for a specific interaction type IT, and $NO_{(IT,MC,t)i}$ is the number of occurrences in the current MAS instantiation of each interaction protocol MC for that IT. This equation is specified in [15].

For the normative knowledge, the acceptance function is defined by Eq. (7):

$$Lac = \left[\frac{Lv * \overline{m} + \overline{P} * m}{2}\right] \tag{7}$$

where *Lac* is the current limit of the parameters defining an interaction protocol (either UL or LL), *Lv* is the previous limit, *m* is the moment and is obtained from the ratio between a constant between zero and one, and the iteration number, $\overline{m}$ is the complement of the moment, i. e., $(1 - m)$, $\overline{P}$ is the average value of the limits of all individuals within 20% from the accepted population. For details see [15].

The *influence function* determines how knowledge of the system influences individuals of the population. For this purpose, the individuals are modified through a targeted mutation operator, in order to change the coordination mechanism in the individual (situational knowledge) or the value ranges of the corresponding parameters (normative knowledge). This is carried out based on a small probability mutation. In [15], the process is specified.

IT	CM	$IO_{(IT,\ CM,\ t-1)}$		$TO_{(IT)}$

Fig. 4. Situational knowledge.

CM	P^1		P^2		...	P^u	
	LL	UL	LL	UL		LL	UL

Fig. 5. Normative knowledge.

3.3. CLEMAS

The cultural learning model is implemented through a computational tool called CLEMAS (Cultural Learning for Multi-Agent Systems) [18]. This tool consists of four main components: (i) the execution engine, (ii) the model of cultural learning, (iii) a graphical interface for configuring the system initially, and (iv) a database that stores the acquired knowledge by individuals in the belief space

The execution engine starts the learning process, by using the initial configuration of the system. This configuration is set through the graphical interface that permit the definition of: (a) the population size and structure of individuals, by specifying their interactions, (b) the interaction types and (c) the interaction protocols for starting the simulation. Besides, the number of generations, the probability of crossover and mutation also stated. Next, the components of the cultural learning model take the loaded data and creates the initial population and the belief space. As the algorithm is in progress, the situational and normative knowledge acquired (belief space) are stored in the knowledge base. In addition, CLEMAS saves a file if we want to use the knowledge in other MAS. For more details on this tool, see [18].

4. Proposal for industrial automation integration based on coordination of MAS by using cultural algorithms

4.1. Process automation based on MAS

Integration necessities in industrial automations makes MAS architectures as suitable approaches that allow the integration of data and services. Specifically, this integration is given, on the one hand, by defining communities of specialized agents in automation tasks, and on the other hand, by the coordination mechanisms that ensure the conversations between agents are achieved optimally. Both characteristics allows the agents orchestrates their communications properly to provide automation services.

Specifically, Intelligent Distributed Automation based on Agents (SADIA, by its acronym in Spanish) is a framework based on MAS, which represents the operations carried out in an industrial environment, from the point of view of process automation [11]. SADIA is a nested architecture, proposing communities of agents that offer services that are scattered in the automation pyramidal architecture ISO/OSI. These agents aim to model logical or even physical objects. SADIA will be used for the analysis of the integration premises in automation by using MAS.

Fig. 6 shows the SADIA architecture, which proposes a structural and functional decomposition to conceive generic functionalities that are required by any production unit. Structural decomposition is in the first abstraction level, where all production units associated with the business objects are represented. Functional decomposition is along the second and third levels of abstraction. Particularly, second level considers all the main functionalities in industrial automation as general services required by any production unit; the third level is devoted to agents offering the services that are required by the agents in the second level. This conception permits the generic modeling of the industrial automation functionalities, viewed as services, through software agents, which are instantiated according to the production unit that such agents are implementing.

At the top level of the architecture, the production process is a MAS, where several production units are modeled as agents. In this level, the agents negotiate with each other to reach agreements

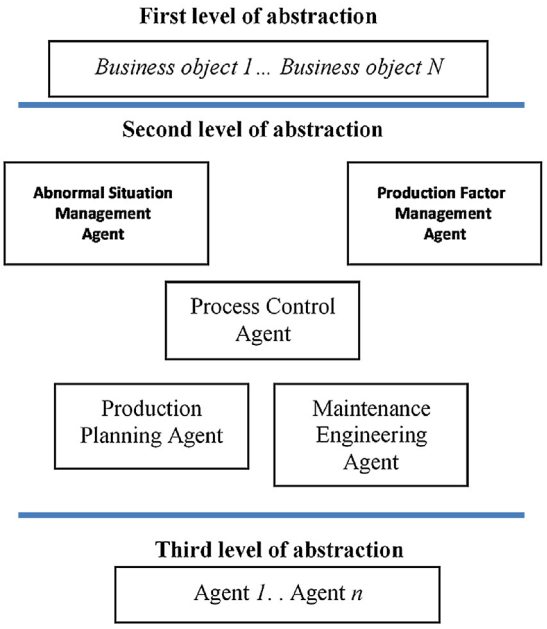

Fig. 6. SADIA architecture.

that would meet production goals; these agreements represent the business logic governing the production process.

The middle level is composed of agents dealing with automation activities of each business object at the top level, named: process control, maintenance engineering, management of abnormal situations, management of production factors, and the production planning. Thus, the above listed activities are common for each agent of the higher level, and consequently, all agents of that level will have a same architecture that consists of agents playing each one of the automation activities. However, there are specific activities associated to specific business objects, they are modeled by specialized agents, which complement the architecture of each top-level agent.

Finally, since the activities carried out by agents of the middle level are complex, a level of innermost abstraction is proposed. The agents of the midlevel are other MAS, distributing the tasks of each activity of automation in the middle level, between several agents. For this lowest level, the framework SCDIA (Distributed Control System based on Intelligent Agents) is used as a reference model of the agents [12]. The SCDIA based agent framework proposes a set of specialized agents that models the automation activities related to physical object of the process, that is, sensors, actuators, controllers such as PLC, HMI's, SCADA, and even communication platforms such as OPC. In this sense, each agent in the lowest level of our agent model for automation is a software abstraction of the physical functionalities of automation components.

Each agent of the SADIA architecture is specified trough five models using MASINA [13]: Agent, Services, Coordination, Communication, and Intelligence. Particularly, Agent and Service models specify all the activities that have been performed by the agents to accomplish their objectives. Coordination and Communication models specify the integration between agents, to implement properly the services offered and required by the agents.

Integration conflicts resolution and complex interaction between agents are specified through such models, and supported by a Services Management Middleware. More details about the models specification can be found in [11,13]. Our work assumes that each agent is properly designed and, particularly, coordination and communication models manage the desired integration. Our approach is focused on the optimization of the coordination mechanisms, specified in the coordination model, through the selection of the proper interaction protocols, which implement the communicative acts between agents present in the model.

From the standpoint of use of intelligent agents, SADIA architecture is implemented in the generic automation architecture using MAS depicted in Fig. 7 [35]. In such figure three functional levels are distinguished, such as: field level, with the actuators and sensors, middle level with the Services Management Middleware called MGS, and the application level where the SADIA agents are spread. The MGS level is composed of a set of software modules that support the agent management, and their interaction with the field devices. MGS is in turn composed of three levels: the Interface Level, the Medium Level, and the Resources Access Level. The interface level is composed of the agents defined by the FIPA specification to support the deployment of agent communities. The medium level is the core of the distributed system, which provides software services that agents require to interact with each other and with the field devices. The Resource Access Level represents the set of services provided by the operating system, including the management of real-time tasks.

Next section presents some premises about how integration arises in MAS-based automation architectures, through the coordination of the different interactions between the agents that compose the MAS.

4.1.1. General premises for integration in MAS-based automation architectures

Several premises are proposed to analyze the problem of integration in the context of study.

4.1.1.1. 1st premise. The agents belonging to different communities in a MAS-based architecture as SADIA, attain the *horizontal*

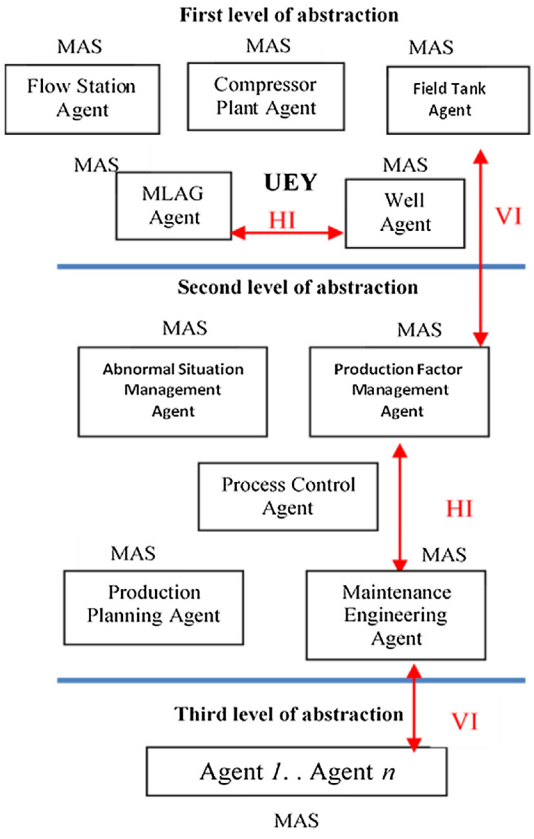

Fig. 8. Interactions between MAS in SADIA architecture.

integration (flat architectures) and *vertical integration* (hierarchical architectures), by coordinating their conversations.

4.1.1.2. 2nd premise. The agents can talk to each other, by sending and receiving data when communication between different agents are ensured, regardless of the abstraction level to which they belong (communities of agents). This enables the *data integration*.

4.1.1.3. 3rd premise. Any model of services-oriented MAS enables orchestrating the execution of services between agents, through the definition of the adequate interaction protocols that define the coordination mechanisms for implementing conversations between agents (e.g., auction or tender), with a minimum of conflict between them. This allows the *service integration*.

4.1.1.4. 4th premise. All the agents that compose the automated system work together, each one performing its specific tasks. This allows the *interoperability between communities.*

In order to illustrate the proposed premises, SADIA is used as the architecture for modeling an oil production process. Suppose that a loop of oil production requires a set of facilities to implement the extraction, treatment, distribution and transport of hydrocarbons [11]. The process of oil production is composed of wells using the technique of gas lift. In this example, the loop of oil production focuses on the exploitation of the deposit unit called UEY, and its main components are: wells, flow stations, field tank, gas compression plants and multiple (next section details this process as a case of study).

The fulfillment of the premises in the SADIA framework is discussed from Fig. 8, as follows:

- Regarding the first premise, SADIA architecture includes both vertical (VI) and horizontal (HI) integration, of the different communities of agents (MAS). Fig. 7 shows the horizontal integration between the agents composing the same level (e.g., between agents representing different business objects at the top level), and a vertical integration among the agents at different levels (e.g., between the field tank agent and mid-level agents, such as the production factor management agent).
- With regard to the second premise, SADIA specifies the interaction of agents, regardless the abstraction level where they belong. This allows the exchange of data between agents that facilitates data integration.
- For the third premise, SADIA is an integration model based on services-oriented MAS. At the top level, the agents associated to the business objects negotiate the services to be provided by other agents, to reach agreements that would meet the production goals.
- Finally, for the fourth premise, a system specified with SADIA ensures interoperability between the different agents of the MAS, through the fulfillment its specific tasks that are invoked as services in conversations describing the dynamic of SADIA.

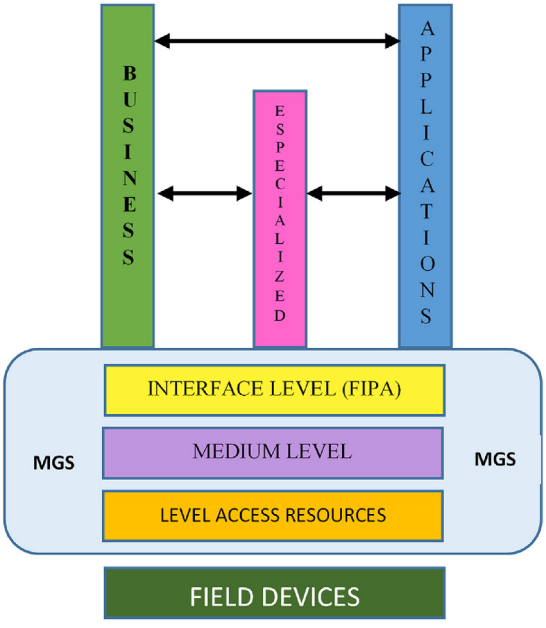

Fig. 7. MAS-based automation Architecture.

4.2. Integration through the MAS coordination

Different interactions exist in a MAS that define the coordination between agents. Our solution approach for integration in automation is managed through the coordination of the interactions between agents. In this sense, in order to achieve the proposed premises, coordination mechanisms in a MAS architecture must have the following characteristics:

- For the first premise, the proper coordination of the interaction between agents, no matter what level/community the agents belong, ensures both, vertical and horizontal integrations. Different interaction protocols can be used properly in the different interactions of a MAS.
- For the second premise, the data integration is considered as implicit fact in the interactions (speech acts) that flow through the communications describing the conversations.
- With regard to the third premise, a good integration of services is guaranteed with the selection of appropriate interaction protocols for the interactions between agents.
- For the fourth premise, the interoperability between communities is ensured, when the agents interact coordinately.

In short, in order to achieve the integration of data and services, through the coordination of the interactions in the MAS based industrial automation architecture, the following steps must be accomplished:

- Define a MAS-based industrial automation architecture.
- Describe the different conversations, sub-conversations and interactions that are needed to meet the different automation tasks types, by using some methodological framework for specifying MAS.
- State the different coordination mechanisms using interaction protocols that will be used in the conversations of MAS.

In the last step, this work proposes the use of an optimization mechanism based on the model of cultural learning presented in the previous section, to select the interaction protocols to be used in the interactions. The learning model will be used as illustrated in Fig. 9. This figure shows a generic example where several agents of the SADIA architecture are performing a conversation. the same conversation can simultaneously have horizontal integration (HI) and vertical integration (VI). The best interaction protocols to implement the interaction types IT_i, IT_j and IT_k, should be proposed by the CA, which select the most appropriate coordination mechanism for the implementation of the conversation.

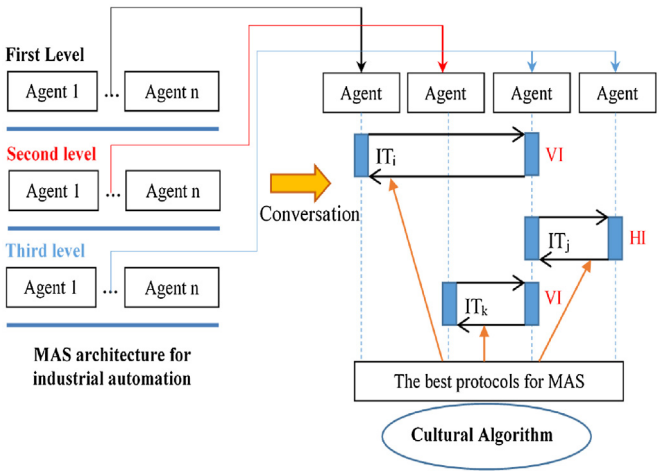

Fig. 9. Optimization of the integration-oriented coordination mechanism of MAS for industrial automation by using CA.

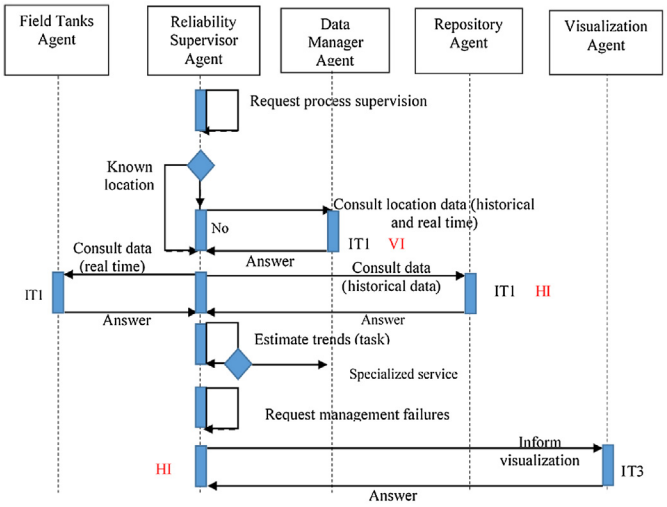

Fig. 10. Interaction diagram of the conversation "Monitoring Process".

5. Case study

The complexity of the real-world examples is in the coordination models of the conversations, where vertical and horizontal integrations are needed in order to offer the services. Our examples have been simplified to show this aspect; in addition, we suppose that the agents are correctly modeled. Our simulation computational tool, called CLEMAS, can propose and evaluate the best set of interaction protocols for implementing the coordination models of the MAS.

5.1. Integration at business objects level

This case study is taken from a real project of an oil industry for which the use of agents was proposed in the process automation to implement smart features in the oil production systems [36].

The level we are interested in this paper is the top level (Fig. 7), because that is the level where the process automation services are offered [35]. This level is composed of two communities of agents [37]. One of them is the community of Business Agents, which describe the different business objects (logical and functional abstractions of real processes). The other community is the Application and Specialized Agents, which consist of all agents that perform specific functions of supervision, control, optimization, visualization, and other specialized agent applications and/or legacy applications [37].

In our case study, a loop of oil production using the technique of gas lift is modeled, which is composed of a set of facilities that are the "Business Object" agents, such as: wells (PZ), flow stations (EF), gas compression plants (PC), multiple of gas lift (mLAG), field tank (PTQ).

5.1.1. Description of the integration context

In the following, several service-oriented scenarios of the oil production process are described.

5.1.1.1. Scenario of the service "monitoring process". This scenario is based on the conversation "monitoring process" whose interactions are presented in Fig. 10. In this scenario, the reliability supervisor agent starts monitoring field tank agent (business object). For this task, the supervisor agent starts the conversation that allows accessing the process data in real time, to estimate the future trends (vertical integration). This agent analyzes the results and generates information in a clear and interpretable way. Additionally, the agent can generate alarms and fault management requests. These results are transmitted to the visualization agent (horizontal integration), to be presented to human operators.

5.1.1.2. Scenario of the service "designing control". This scenario is based on the conversation "designing control", illustrated in Fig. 11. In this case, the well agent requests to the control designer agent the design of a control strategy (horizontal integration). The designer control agent must locate the data (required to the well agent and data manager agent) (vertical integration), and then it determines the control strategy, plans the control, and finally, transmits the information to the well agent (horizontal integration).

5.1.1.3. Scenario of the service "evaluating performance of the control plan". This scenario is based on the conversation "evaluating performance of the control plan", which is shown in Fig. 12. The flow station agent asks to the control evaluator agent for conducting a performance evaluation of the control plan (horizontal integration). The control evaluator agent requests current control data to the flow station agent, and historical control

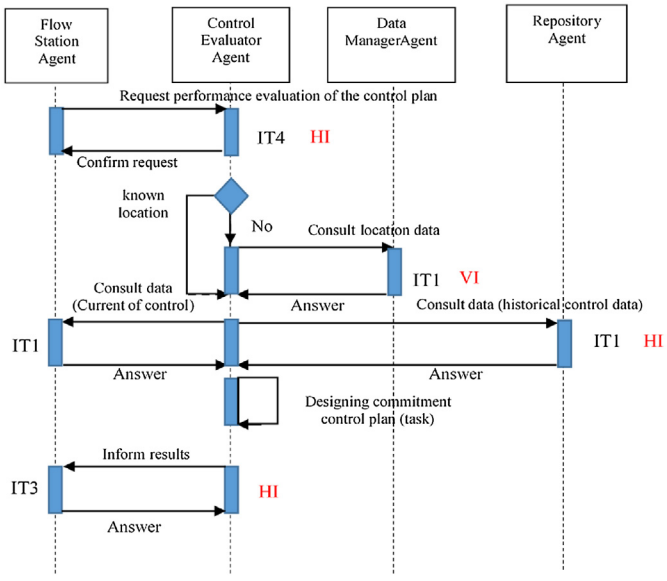

Fig. 11. Interaction diagram of the conversation "Designing Control".

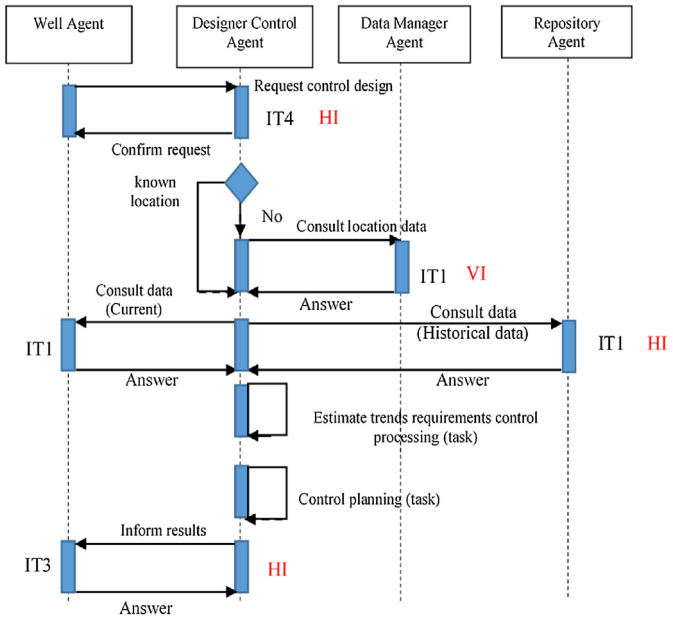

Fig. 12. Interaction diagram of the conversation "Evaluating Performance of Control Plan".

data to the repository agent. When the task is completed, the agent transmits the results to the flow station agent (horizontal integration).

5.1.2. Experimental analysis using CLEMAS, for the integration using the MAS coordination

In the cultural model proposed in Section 3.2, each individual in the population is seen as a MAS. In this case, it consists of the three conversations given above (Figs. 10–12). At the beginning of the simulation, an interaction protocol is assigned to each IT, which can be auction or tender. The parameters of the evolutive process are:

- Population: 50 individuals
- Number of generations: 65
- Crossover probability: 0.7
- Mutation probability: 0.07

The results provided by CLEMAS are shown in Table 1. The protocol with the highest number of occurrences was tender, with 55.01%. The table does not show results for IT2, because that conversation is not included. In case of IT1 (consult), tender was the more chosen protocol, the Dutch auction was selected for IT3 (inform), and for IT4 (request) tender was selected.

With regard to the compliance of the premises proposed in Section 4.2, we can conclude:

Both horizontal and vertical integration are presented in the different interactions shown in the case study, such as in the case of conversation "process monitoring". In the first IT1, a vertical integration (IV) is presented, because the reliability supervisor agent (top level) communicates with the data manager, which belongs to the MGS level (lower level). For the following IT1 and IT3, a horizontal integration (IH) is given because the reliability supervisor agent, the field tank agent, the repository agent, and the visualization agent belong to the same level.

Table 1
Results (Percentage of use of each coordination mechanism in each IT, based on 20% of the population).

IT	Tender	English Auction	Dutch Auction
Consult (IT1)	55,36%	5,20%	37,90%
Assign (IT2)	0,0%	0,0%	0,0%
Inform (IT3)	30,22%	33,18%	34,80%
Request (IT4)	90,39%	5,21%	1,32%
Total of occurrences	55,01%	12,55%	30,32%

In the interaction diagrams presented, the exchanges of message between individuals can be observed. In particular, conversations are composed of speech acts (communicative interactions) between agents. Speech acts reveal data integration present at the conversations.

Interaction protocols (auction and tender) allow planning the execution of all services requested in the conversation. For example, the interaction diagram in Fig. 12 shows how the designer control agent offers services to determine the control strategy and to plan the control, for the well agent. CLEMAS determines which interaction protocol should be used in every conversation present in the MAS, to ensure optimal integration of services.

Finally, in this case study there are several agent communities at different levels. Their conversations have interactions between agents on the same level, or between agents of different levels. The optimization that makes CLEMAS, allows solving in an implicit manner the problem of integration, and the overall interoperability between all agents in the MAS, regardless of the levels/communities where they are (belong).

5.2. Integration in industrial supervision processes, for fault management tasks

This case study is related to a System of Fault Management (SMF), which is composed of two modules: the first one performs tasks of monitoring and analysis of the failure, and the second one provides the maintenance tasks [38]. The SMF interacts with the maintenance engineering unit to get information related to productivity rates of the process, resource management, among others, and with the fault-tolerant controlled process. The module for monitoring and failure analysis includes the tasks of fault detection and diagnosis; the module for supporting the maintenance tasks includes the prediction of the occurrence of a functional failure, the planning of preventive maintenance, and the maintenance execution [38].

The SMF is a system composed of intelligent agents which can cooperate for solving problems related to the management of system failures, as illustrated in Fig. 13. Moreover, some activities of the SMF should be adjusted to a distributed computing model, such as those ones performed for detecting failure in devices or processes, the estimation of performance index, among others. For the MAS specification, the methodology MASINA [13,19] was used, which provides several models for agent specifications [38].

The SMF provides the following services: monitoring, detection, location, analyzing, and predicting the occurrence of a fault. Maintenance actions on the control system are also considered as services.

The MAS of a SMF is based on the model SCDIA [12]. The SCDIA has been adapted to the problem of fault management, which is viewed as a generic problem of closed loop control. Therefore, eight agents were defined: five specialized agents, such as Detector Agent, Locator Agent, Diagnostician Agent, and Predictor Agent, the rest of the agents are Coordinator Agent, Controller Agent, Actuator Agent, and Observer Agent. Specialized, Controller and Coordinator agents are at the Supervisor level, and the Observer and Actuator agents are at the level of process.

Through the coordination model of MASINA, six conversations are described: on-condition maintenance, maintenance tasks, urgent tasks, re-planning of tasks, maintenance status, and identification of functional failures. The detailed specification of

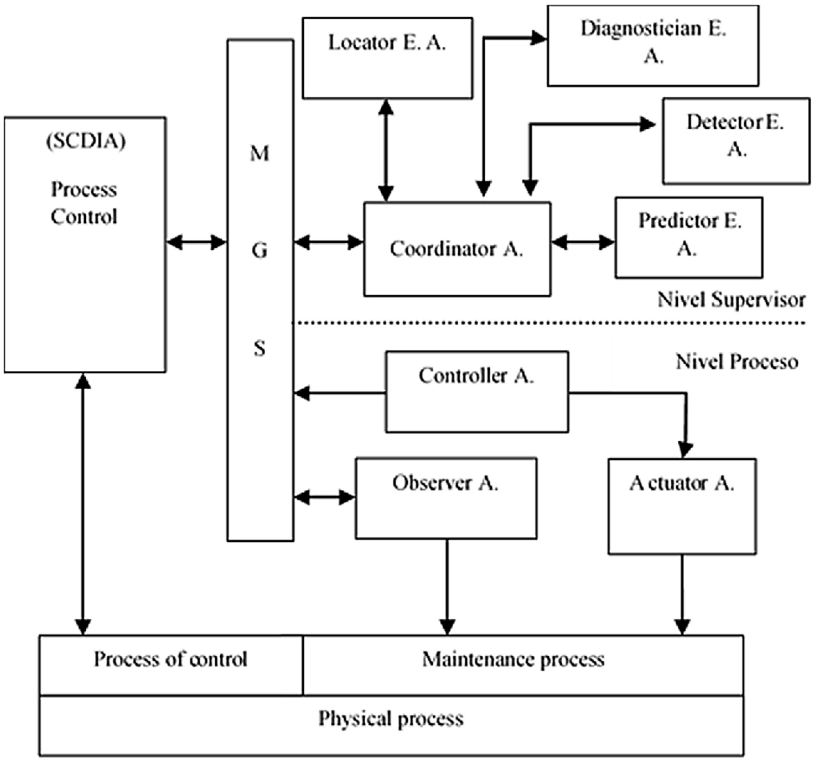

Fig. 13. MAS of the SMF.

these agents, their conversations, and interactions, are detailed in [38].

5.2.1. Description of the integration context

In particular, we analyze the conversations "re-planning tasks" and "maintenance status", respectively.

5.2.1.1. Scenario of the service "Re-planning tasks". this scenario is based on the conversation "re-planning tasks". Through this conversation, the coordinator agent requests information to the database agent to reschedule expected maintenance tasks on the system, and making a new maintenance plan (horizontal integration). If the task is urgent and it cannot be rescheduled, an alarm is given (vertical integration). In Fig. 14, the interaction diagram of this conversation is shown.

5.2.1.2. Scenario of the service "maintenance status". This scenario describes the conversation "maintenance status". In Fig. 15, the interaction diagram of this conversation is shown. Through this conversation, the observer agent consults information from the database and the actuator agent (horizontal integration), in order

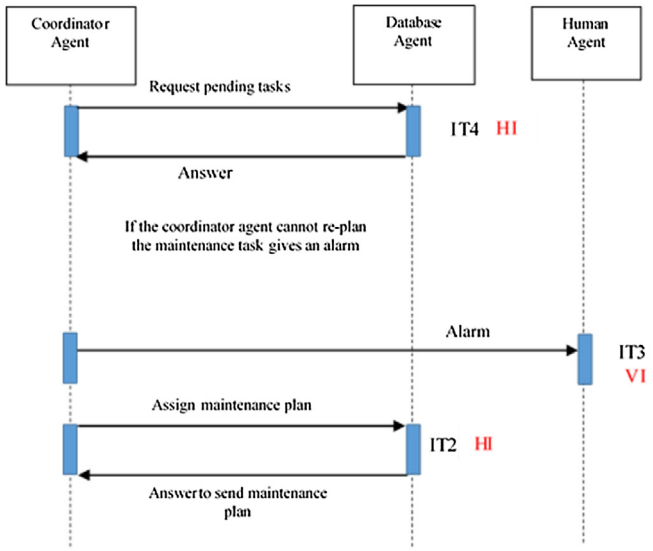

Fig. 14. Interaction diagram of the Conversation "Re-planning of Tasks".

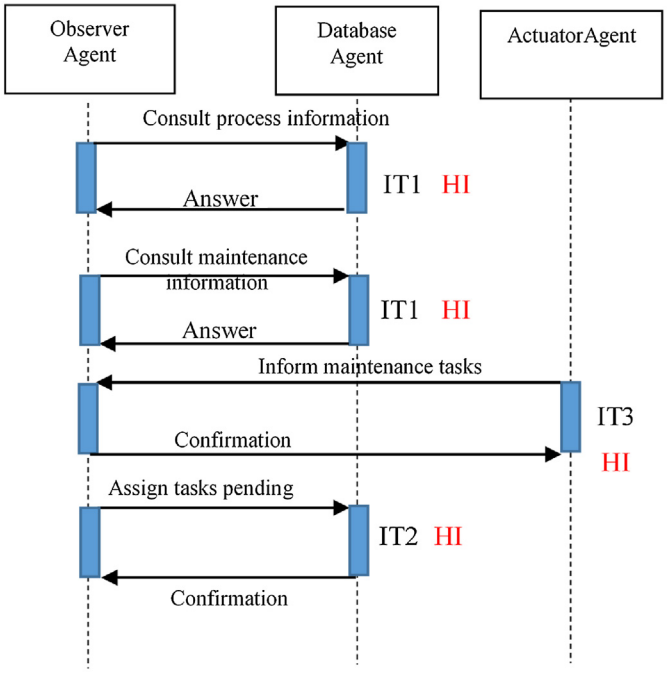

Fig. 15. Interaction diagram of the conversation "Maintenance Status".

to determine the status of maintenance tasks in the system. In addition, the maintenance tasks that are expecting can be re-allocated (horizontal integration).

5.2.2. Experimental analysis using CLEMAS, for the integration using MAS coordination

We assume that in the conversation "Re-planning tasks", the sub-conversation IT2 is only optimized. For accomplished this sub-conversation, we suppose that there are four available agents (3 database agents and one coordinating agent). In case of conversation "Maintenance Status", the sub-conversation IT2 and IT3 are only optimized, and we suppose that there are six available agents (one observer agent, 3 database agents and 2 actuator agents).

The maximum number of rounds was set in 5 for the auction, and for tender just 1. The initial values of the parameters for each the mechanisms are:(i) for auction: $C_0 = [5 \ldots 15]$ $\varepsilon = [5 \ldots 20]$, $C_P(j) = [1 \ldots 5]$, (ii) for tender, $M(F) = [1 \ldots 3]$, and $f(T) = [5 \ldots 20]$. For this simulation, the parameters of the evolutive process are:

- The population is of 20 individuals
- 50 generations
- Crossover probability is 0.7
- Mutation probability 0.5.

Table 2 shows the results of optimizing interaction IT2 and IT3 for conversations "Re-planning tasks" and "Maintenance status", respectively. Results shows that the tender has prevailed in 41.02% of cases, compared to other protocols (total occurrences). The percentage of use of this interaction protocols is 74.20% for IT2 and 16.04% for IT3. The English auction is the least used, only 18.57%.

With regard the premises proposed in Section 4.2, we can conclude:

- The conversations "re-planning tasks" and "maintenance status" require the interaction between different levels of the system. A vertical integration occurs by interacting the coordinator agent with the database agent (see Fig. 14). A horizontal integration occurs by interacting the observer and actuator agents (see Fig. 15).
- There is a flow of data in both conversations, regarding search for pending tasks and the definition of the maintenance plans. As well as in the transmission of process information and the report of the maintenance tasks. Then, in these conversations perform data integration.
- Specialized agents provide services to other agents. The conversations show the integration of services, to perform the tasks of re-planning and maintenance.
- Interoperability is shown in both conversations. Particularly, when the coordinator agent sends the maintenance plans to the database agent and the actuator agent in conversation "Re-planning task", and when the observer agent stores the remaining tasks in the database agent in conversation "Maintenance Status". CLEMAS determines which interaction protocols should be used in every interaction, to implement the adequate integration.

Table 2
Results (Percentage of use of each interaction protocols in each IT, based on 20% of the population).

IT	Tender	English Auction	Dutch Auction
Consult (IT1)	0,0%	0,0%	0,0%
Assign (IT2)	74,20%	2,60%	22,18%
Inform (IT3)	16,04%	35,13%	68,02%
Request (TI4)	0,0%	0,0%	0,0%
Total de occurrences	41,02%	18,57%	40,10%

5.3. Qualitative comparison of integration models

Qualitative comparison of our approach, with other works proposing agents in automation architectures, are described in Table 3. Some criteria are considered: the type of integration, the paradigm of implementation used for the integration, and the integration objectives.

The Table 3 shows different approaches. The first aspect to notice is the variety in the types of integration. Our approach is based on a service-oriented architecture, as is proposed in [7,8]. In [10] a service-oriented MAS based integration is proposed, while the rest of works propose integrations based on ontologies, or through intelligent control mechanisms. The implementation paradigm is described in terms of how this integration is achieved. In our approach, it is achieved through a collective learning in the MAS by using CA. In other works, they use web services or ontological standards for integration. Finally, the integration goal in our case, is to achieve interoperability, while the other works aim to semantic integration, system reconfiguration, or the coordination of behavior.

The types of integration in the presented works are four: SOA, SOI (for agents and MAS), ontological, and intelligent. According to the definition of SOI, it is an integration type that is part of SOA. SOI manages the integration problems of the inflexible legacy systems and heterogeneous systems, in order to enable the functionality of reusable services in heterogeneous applications. This is achieved by using agents (SOI) or communities of agents (SoMAS). SOA as architecture allows setting goals that can go beyond those ones proposed by SOI, because it allows the two categories of SOI. Ontological integration is also known as semantic integration, and covers several disciplines such as database, information and ontologies integration, i.e., it is a data-oriented integration. This integration is subject to the limitation that must be modeled individually for each of the data sources as ontologies, leading to three forms of integration: simple, multiple or hybrid ontology. Finally, our approach proposes an integration based on SOA, and it allows the horizontal and vertical integration, as also the data and service integration. In addition, our approach can be considered

Table 3
Comparative table of models of integration.

Models	Integration type	Implementation paradigm	Objective
Model proposed in this work	SOA	SOA through collective learning of coordination mechanisms (interaction protocols)	Interoperability between agents of MAS
[8]	SOA	SOA through web services	Automated reconfiguration
[10]	SoMAS	SOI through MAS	Flexibility, autonomy and reconfiguration
[7]	SOI	SOI through agents	Promote integration into a network of virtual companies
[26]	MAS coordination architecture based on ontology	Based on the criterion standard ontological CIM/XML	Ontological integration
[31]	Intelligent integration	Through local intelligent control and global intelligent control	Coordination of multi-objective behaviors

within the intelligent integration since it is based on the artificial intelligence, through the coordination of MAS conversations by using a collective learning process.

6. Conclusions

The area of industrial production requires data, information and knowledge, that are distributed throughout the company or other external environments of the company. Basically, they are associated with the installed production capacity, and the external conditions are associated with market requirements.

MAS are presented as a paradigm to support the rapid growth of industrial processes with high requirements of integrated actions. Within these processes arises industrial automation, which is increasingly complex and requires the integration of different systems and applications.

The different paradigms of integration, such as those ones based on services, data, among others, allow fulfilling necessities that arise in the automation systems. In particular, the agent-based automation architecture, as proposed in this paper, allows the vertical and horizontal integration, the services and data integration, and ultimately, the interoperability of their agent communities. They enable characteristics of flexibility, autonomy and reconfiguration, in the industrial processes.

Specifically, the problem of integration in industrial automation generally relies in the difficulty of coordination between the processes that compose the automation architecture. In this work, we propose an approach from the optimization of the coordination mechanisms (interaction protocols), for implementing the different conversations in the MAS-based automation architecture. For this purpose, we develop a model of collective learning based in CA. The optimization model uses the formal models of auction and tender as interaction protocols, which consider the variables and parameters of the communication process between agents. Results of the optimization process using CLEMAS show that adequate coordination mechanism can be selected based on an objective function defined by processing and communication costs.

Acknowledgments

Dr. Aguilar and Dr. Cerrada have been partially supported by the Prometeo Project of the Ministry of Higher Education, Science, Technology and Innovation of the Republic of Ecuador.

References

[1] J. Aguilar, A. Rios, F. Hidrobo, M. Cerrada, Sistemas Multiagentes y sus Aplicaciones en Automatización Industrial, Talleres Gráficos Universitarios, Mérida: Universidad de Los Andes, 2012.
[2] Lenzerini, M. (2002). Data integration: A theoretical perspective. Tutorial at PODS 2002, Madison, Wisconsin, USA
[3] B. Christudas, Service Oriented Java Business Integration, Packt Publishing, 2008.
[4] L. Ribeiro, J. Barata, P. y Mendes, MAS and SOA: complementary automation paradigms, IFIP—Int. Fed. Inf. Process. 266 (2008) 259–268.
[5] R. Sesseler, Building agents for service provisioning out of components, In Proceedings of the Fifth International Conference on Autonomous Agents (2017) 218–219.
[6] X. Nguyen, R. Kowalczyc, Enabling agent-based management of web services with WS2JADE, In Proceedings of the Fifth International Conference on Quality Software (2005) 407–412.
[7] S. Weiming, H. Qi, W. Shuying, L. Yinsheng, G. Hamada, An agent-based service-oriented integration architecture for collaborative intelligent manufacturing, Rob. Comput. Integr. Manuf. 23 (3) (2007) 315–325.
[8] H. Villaseñor, A. Bepperling, A. Lobov, H. Smit, A. Colombo, L. Martinez, Integration of multi-agent systems and service-oriented architecture for industrial automation, In 6th IEEE International Conference on Industrial Informatics (2008) 768–773.
[9] L. Na, Z. Weimin, W. Feng, Y. Zhencheng, Q. Feng, An agent-based service-oriented integration architecture for chemical process automation, Chin. J. Chem. Eng. 23 (1) (2015) 173–180.
[10] J. Mendes, P. Leitao, F. Restivo, A. Colombo, Service-oriented agents for collaborative industrial automation and production systems, Lect. Notes Comput. Sci. 5696 (2009) 13–24.
[11] C. Bravo, J. Aguilar, F. Rivas, Diseño de una arquitectura de automatización industrial basada en sistemas multi-agentes, Ciencia e Ingeniería 25 (2) (2004) 75–88.
[12] J. Aguilar, M. Cerrada, G. Mousalli, F. Rivas, F. Hidrobo, A multiagent model for distributed control systems. In: knowledge-based intelligent information and engineering systems, Lect. Notes Comput. Sci. 3681 (2005) 191–197.
[13] J. Aguilar, I. Besembel, M. Cerrada, F. Hidrobo, F. Narciso, Una Metodología para el Modelado de Sistemas de Ingeniería Orientado a Agentes. Iberoamerican, J. Artif. Intell. 12 (38) (2008) 39–60.
[14] Reynolds, R. (1999). Cultural Algorithms: Theory and applications. New ideas in optimization. David Corne, Marco Dorigo, and Fred Glover. Editors, chapter twenty-four: 367–377.
[15] J. Terán, J. Aguilar, M. Cerrada, Collective learning in multi-agent systems based on cultural algorithms, CLEI Electron. J. 17 (2) (2014).
[16] J. Ferber, Les Systèmes Multi-agents: Vers Une Intelligence Collective, InterEditions, 1995.
[17] J. Terán, J. Aguilar, M. Cerrada, Mathematical models of coordination mechanisms in multi-agent systems, CLEI Electron. J. 16 (2013) 1125–1137.
[18] J. Terán, J. Aguilar, M. Cerrada, Cultural learning for multi-agent system and its application to fault management, IEEE World Congr. Comput. Intell. (2014) 2188–2195.
[19] J. Aguilar, M. Cerrada, F. Hidrobo, A methodology to specify multiagent systems, Lecture Notes Artif. Intell. 4496 (2007) 92–101.
[20] FIPA Abstract Architecture Specification. Standard of the Foundation for Intelligent Physical Agents. (2002).
[21] Foundation for Intelligent Physical Agent, (2002). FIPA Communicative Act Library Specification. Document number XC00025E.
[22] H. Nwuana, L. Lee, N. Jennings, Co-ordination in software agent systems, BT Technol J. 14 (1996) N. 4.
[23] H. Bin, H. Ding, Z. Guan, Hybrid subgroup coordination of multi-agent systems via nonidentical information exchange, Neurocomputing 168 (2015) 646–654.
[24] T. Moser, M. Merdan, A pattern-based coordination and test framework for multi-agente simulation of production automation systems, In Eighth International Conference on Complex, Intelligent and Software Intensive Systems (2010) 526–533.
[25] K. Nagorny, A.W. Colombo, U. Schmidtmann, A service-and multi-agent-oriented manufacturing automation architecture: an IEC 62264 level 2 compliant implementation, Comput. Ind. 63 (8) (2012) 813–823.
[26] L. Liu, X. Zu, R. Xu, Multi-agent system coordination architecture and its use in electric power decision support system, In 2008 6th IEEE International Conference on Industrial Informatics (2008) 731–736.
[27] R. Ostos, F. Ramos, B. Cartillo, V. Félix, Selection of coordination mechanisms in intelligent environments, Latin Am. Trans. IEEE 13 (9) (2015) 3120–3126.
[28] D.X. He, G.H. Xu, Z.H. Guan, M. Chi, D.F. Zheng, Hybrid coordination of multi-agent networks with hierarchical leaders? Commun. Nonlin. Sci. Numer. Simulat. 27 (1) (2015) 110–119.
[29] K. Hadeli, P. Valckenaers, M. Kollingbaum, H. Van Brussel, Multi-agent coordination and control using stigmergy, Comput. Ind. 53 (1) (2004) 75–96.
[30] P. Valckenaers, K. Hadeli, B. Saint Germain, P. Verstraete, H. Van Brussel, MAS coordination and control based on stigmergy, Comput. Ind. 58 (7) (2007) 621–629.
[31] H. Van Brussel, J. Wyns, P. Valckenaers, L. Bongaerts, P. Peeters, Reference architecture for holonic manufacturing systems: PROSA, Comput. Ind. 37 (3) (1998) 255–274.
[32] N. Kubota, N. Aizawa, Intelligent control of multi-agent system based on multi-objective behavior coordination, In IEEE World Congress on Computational Intelligence (2008) 1458–1463.
[33] Valckenaers, P., Van Brussel, H., (2016). Design for the unexpected Butterworth-Heinemann Publisher.
[34] Foundation for Intelligent Physical Agent, (2002). FIPA Interaction Protocol Library Specification. Document number SC00037J.
[35] F. Hidrobo, A. Rios, J. Aguilar, L. Leon, An architecture for industrial automation based on intelligent agents, WSEAS Trans. Comput. 4 (12) (2005) 1808–1815.
[36] F. Martínez, J. Aguilar, C. Bravo, Planificación en Automatización basado en Sistemas Multiagente, Rev. Avances Sistemas Inform. 8 (2011) 959–966.
[37] A. Ríos-Bolívar, F. Hidrobo, M. Cerrada, J. Aguilar, Control and supervision system development with intelligent agents, WSEAS Trans. Syst. 6 (1) (2007) 141–148.
[38] M. Cerrada, J. Aguilar, J. Cardillo, R. Faneite, Agents-based design for fault management systems in industrial processes, Comput. Ind. 58 (2007) 313–328.