Artificial cognitive control with self-x capabilities: A case study of a micro-manufacturing process

Rodolfo E. Haber*, Carmelo Juanes, Raúl del Toro, Gerardo Beruvides

ABSTRACT

Nowadays, even though cognitive control architectures form an important area of research, there are many constraints on the broad application of cognitive control at an industrial level and very few systematic approaches truly inspired by biological processes, from the perspective of control engineering. Thus, our main purpose here is the emulation of human socio-cognitive skills, so as to approach control engineering problems in an effective way at an industrial level. The artificial cognitive control architecture that we propose, based on the *shared circuits model* of socio-cognitive skills, seeks to overcome limitations from the perspectives of computer science, neuroscience and systems engineering. The design and implementation of artificial cognitive control architecture is focused on four key areas: (i) self-optimization and self-leaning capabilities by estimation of distribution and reinforcement-learning mechanisms; (ii) portability and scalability based on low-cost computing platforms; (iii) connectivity based on middleware; and (iv) model-driven approaches. The results of simulation and real-time application to force control of micro-manufacturing processes are presented as a proof of concept. The proof of concept of force control yields good transient responses, short settling times and acceptable steady-state error. The artificial cognitive control architecture built into a low-cost computing platform demonstrates the suitability of its implementation in an industrial setup.

1. Introduction

There is an abundant literature on artificial cognitive architectures in the fields of sensory motor control and robotics, although the actual application of artificial cognitive architectures in industry is still embryonic. Hybrid cognitive architecture that relies on the integration of emergent and cognitivist approaches using evolutionary strategies is proposed in [1] with a cognitive level controlled by artificial immune systems based on genetic algorithms. Bannat et al. [2] presented a seminal paper on how artificial cognition can be applied in production systems. The authors noted that self-optimizing and self-learning control systems are a crucial factor for cognitive systems and identified important gaps such as the individual worker internal model. Sanchez-Boza et al. [3] proposed an artificial cognitive control architecture based on the shared circuit model (SCM). Its main drawback is a lack of systematic procedures for learning and optimization in the proposed five-layer architecture.

The way in which neuro-physiological mechanisms that reinforce learning and cognitive control are integrated in the brain to produce efficient behavior has yet to be understood with sufficient clarity for effective systems to be modeled [4]. Nevertheless, reinforcement learning has been explored in artificial cognitive control by means of computational models to control robotic systems [5]. Recent studies corroborate what has been known for a long time: automatic and flexible decision-making procedures are the cornerstone to reduce human intervention in the presence of complexity, uncertainty, background noise, and large data volumes typical of production systems [6,7]. Recent investigations have also shown how feedback information provides data on the environment and the system to the cognitive controller, which it needs to activate Q-learning and dynamic optimization in the cognitive tracking of radar benchmarks [8].

New initiatives are now emerging that apply the results of investigations in the field of artificial cognitive systems in response to specific challenges in industry and services [8]. Recent results in disciplines such as the neurosciences, psychology, artificial intelligence, robotics and other studies related to new machines and intelligent processes have begun to approach the foundation of

a computational theory of intelligence [9,10]. Thus, the main purpose of this study is to emulate human socio-cognitive skills, so as to approach control engineering problems in an effective way at an industrial level. An integrated cognitive architecture from a control perspective can be defined as a system that is able to reproduce all aspects of behavior, while remaining constant across different domains and knowledge bases [11,12]. Integrated cognitive architectures that seek to imitate the major capabilities of human intelligence have been used to explain a wide spectrum of human behavior [13]. Moreover, numerous publications reflect the current pace of its progress in the field of cognitive science, all of which cannot be summarized in the context of the present study [14].

Nevertheless, all of that research is based on the role of internal (direct and inverse) models in cognitive tasks. From a physiological point of view, the connection between the paradigm of internal control and brain-cerebellum connectivity has been advanced as a basis for explaining human intelligence [15]. Research has corroborated this link as a key component of human intelligence from a functional point of view [16]. Moreover, the use of internal models to explain some socio-cognitive skills based on human experience is evident from a psychological point of view [17].

Despite the importance of cognitive architectures as a research area, strategies for the application of artificial cognitive control at an industrial level have many constraints and there are very few formal reviews on control engineering [18,19]. Moreover, relevant aspects of cognitive control architectures have yet to be sufficiently well addressed: firstly, self-learning and self-optimization based on interaction; secondly, procedures for assessing cognitive architectures are limited and their availability is often restricted. Several cognitive architectures are used in many applications, although their implementations with few exceptions are not publicly available. Assessment of their evaluation criteria and performance indices is therefore not easy for control engineering and computers in industry. Such a task would require associating and defining figures of merit related to transient behavior, dynamic and steady state systems, and control effort, among others, all of which hinders any comparison of the present-day capabilities and the performance of these architectures. Finally, many cognitive architectures lack biological inspiration. It is essential that computational implementation of architectures have both biological and psychological roots in real applications. Computational architectures are at present somewhat limited to cognitive "psychological" validity. The architecture presented in this paper is inspired in neuroscience [20] in conjunction with control engineering strategies and methods [21].

Micro-scale manufacturing is a clear example of a dynamic system operating in an environment characterized by continuous change, being a perfect stage to proof new cognitive control strategy. In this scenario, one of the main objectives is the development of technologies and algorithms that enable faster, self-organized, self-optimized behavior process control systems. These manufacturing processes are characterized by the presence of nonlinear and time-variant dynamics that emerge from the behavior of temperature, forces, torques and other representative variables; characteristics that increase the functional complexity of micromanufacturing and the functional requirements and precision of sensors, actuators and computing resources [22,23].

In this study, we describe artificial cognitive control architecture with self-optimization and self-leaning capabilities and its simulation and real-time application to the force control of micro manufacturing processes as a proof of concept. The architecture, based on the model of socio-cognitive skills, overcomes the limitations of the neuroscientific approach [24–26] and takes the principles of simplicity and scalability into account. A further challenge is to implement the architecture in a portable programming language for its assessment and validation in simulated and real micro-manufacturing processes.

To the best of the authors' knowledge, the main contributions of this paper rely on three main pillars. While most cognitive architectures include new systems of perception, the first pillar of this computational architecture in this paper, inspired and fed by recent progress in neuroscience, is human-like perception and real-time interaction with the environment. The second pillar is related to the design and implementation of self-learning and self-optimization capabilities for industrial computing. The architecture introduces specific methods for reinforcement learning and heuristic optimization. The third pillar is built around the assessment computational and cost suitability of the cognitive architecture. Its implementation in a low-cost computational platform aims to facilitate public domain availability and technology transfer in industry.

Following this introduction, the paper has the following structure: a brief review of certain artificial cognitive architectures is presented in Section 2. Mechanisms for enabling self-optimization and learning in the artificial cognitive architecture are then presented in Section 3. The design concepts and implementation procedures of the proposed artificial cognitive architecture are described in Section 4. Simulation and results in a real application to a microdrilling processes are shown in Section 5. Finally, some concluding remarks on the experimental results and future investigations are outlined in Section 6.

2. Artificial cognitive architectures

It would be difficult to summarize and review all the well-known cognitive architectures, such as BDI [27], ICARUS [28], SOAR [29], CLARION [30,31], PASAR [32], and LIDA [33], among others, which control a number of neuro-computational mechanisms [34]. Different criteria such as properties and features, agent capacities, factors in the environment, generality, psychological validity and effectiveness have in various cases formed the basis for their comparisons. Vernon et al. [35] conducted a review of various cognitive architectures such as SOAR, ICARUS, ACT-R and others, which was limited to an analysis of relevant design aspects.

The artificial cognitive control architecture that we propose, based on the *shared circuits model* of socio-cognitive skills [17], seeks to overcome limitations from the perspectives of computer science, neuroscience and systems engineering. Sánchez-Boza et al. [36] reported an initial attempt to design an artificial cognitive control system, although with two main limitations: a lack of specific procedures for enabling self-capacities such as self-optimization and learning and non-generalizable computational systems that could be deployed on low-cost computing platforms.

The SCM approach is supported on a layered structure that reflects socio-cognitive skills (i.e., imitation, deliberation, and mindreading) by means of control mechanisms such as mirroring, and simulation. Basically, SCM is based on the observation of the human brain. The first kind of behavior is covered by the action of SCM layer 1, while the behavior described in the forward model is covered by SCM layer 2. Layer 4 of the scheme is in charge of controlling when one type of behavior or another should be performed.

A further behavior is imitation that, in addition to playing an important role in both human sociability and development, are a means of learning. Imitative learning takes place when mirroring the actions of others in response to the circumstances. The observer first copies previously observed input/output associations, in order to perform this task, which inhibits the mirroring mechanism. SCM represents this mirroring capacity in its layer 3. The interaction between layer 3 and the inhibition control

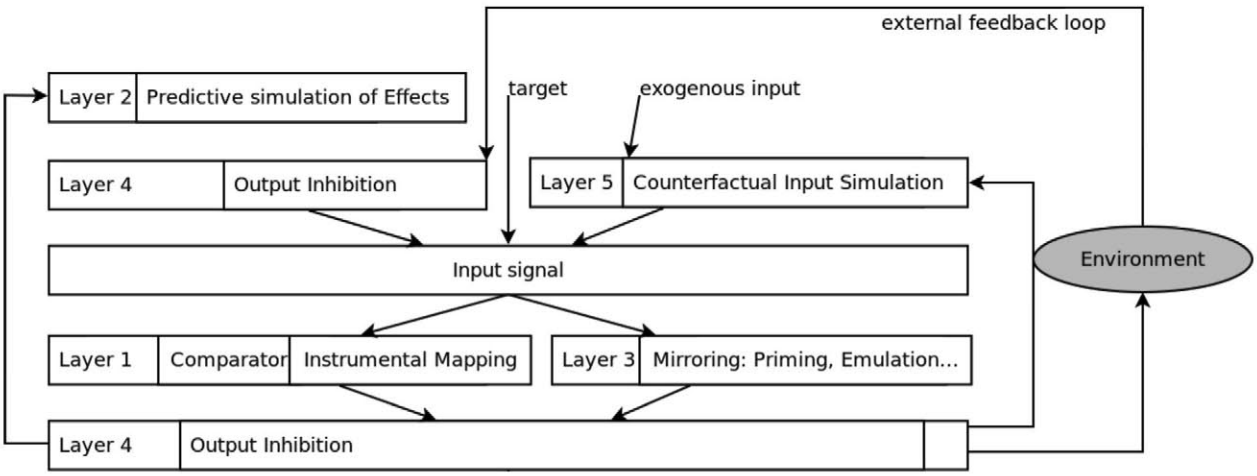

Fig. 1. Conceptual scheme of the shared circuits model approach based on [17].

performed by layer 4 serves to emulate the agent's capability to distinguish self from others.

SCM also describes, from a functional point of view, the way in which the agent can carry out the cognitive skill of mindreading. This capacity is emulated by the operation of layer 5, which is in charge of simulating other possible related inputs that are external (exogenous) to the agent. A layer-based scheme of SCM is depicted in Fig. 1.

A modified shared circuits model (MSCM) based on Hurley's work is proposed in [17]. Five modules were constructed made up of one or more processes performed by the SCM layers. The MSCM proposal defines each module in terms of an emulative cognitive ability. MSCM embodied a computational infrastructure that is plausible from a neuroscientific and psychological perspective, but which lacks a generalizable approach with optimization and learning mechanisms. More details about the five modules and the overall performance can be found in [3]. The main drawbacks are:

- A tailored design of the architecture without a systematic methodology means that it is not extendable to other types of processes or even to other execution configurations.
- A lack of computational strategies to enable self-optimization and learning. These strategies improve the performance of the artificial cognitive control system facing different situations.
- Module-driven architecture is mapped from Hurley's layer concept, but is solely based on a single type of model. For instance, only fuzzy models can be used in the *single loop* configuration.

3. Mechanisms for learning and self-optimization

In this section, we present the algorithms that we have used to enable the auto-optimization and learning. We then describe the original algorithms on which they are based and the modifications or specific assumptions introduced to re-design them.

Reinforcement learning belongs to a category of unsupervised learning techniques [37]. It is a learning paradigm with learning by rewards/penalties with some interesting applications for controlling complex systems, so as to maximize numerical performance measures that express a long-term objective. The analysis of all available reinforcement learning methods is beyond the scope of this paper, although [38] offers a fairly comprehensive catalog of learning problems with a description of an important number of state-of-the-art algorithms.

This work is centered on the Q-learning algorithm, which is a model-free reinforcement learning technique. The main rationale behind this choice is the simplicity of its approach, its model-free feature and the good results of this algorithm reported in the literature. Q-learning can be used to find an optimal action-selection policy for any given (finite) Markov decision process. It performs by learning an action-value function that ultimately generates the expected utility of taking a given action in a given state and it then follows the optimal policy. When such an action-value function is learned, the optimal policy can be constructed by simply selecting the action with the highest value in each state. Additionally, Q-learning can handle problems with stochastic transitions and rewards, with no further adaptation.

The literature is very rich with a wide range of deterministic and stochastic methods for solving optimization problems. Many optimization methods can be applied for this task ranging from genetic algorithms to particle swarm optimization [39,40]. Optimal tuning of the parameters, rather than the optimization of the structure or topology, is adopted in the cognitive architecture within the broad possibilities of optimization, which is computationally simpler and sometimes brings better results than nonlinear systems [41]. One of the main applications of these techniques is the optimal setting of controller parameters (scaling factors or gains) for non-trivial and sometimes intractable tasks [42].

Evolutionary algorithms (EAs) have demonstrated their suitability as a method for multiobjective optimization. EAs maintain a family of solutions during the optimization process, which have the potential to store a set of simultaneous trade-off solutions with the potential to exploit the synergies of a parallel search across all possible solutions. However, EAs are usually experimentally evaluated using various test problems, because an analytical assessment of their behavior is very complicated. Thus, their performance on random problems cannot be guaranteed prior to application [43,44].

The optimal setting of fuzzy controller strategies based on stochastic gradient-based optimization is reported in different works [45–47]. However, many of these optimization techniques have yet to be applied in real industrial processes, due to the high complexity of optimization algorithms, the need to define appropriate cost functions and performance indices, appropriate behavior and/or the lack of empirical formulas for use in industry.

The estimation of distribution algorithms (EDAs) has emerged in the middle ground between Monte-Carlo simulation and EAs. In EDAs, a probabilistic model is built, based on elite individuals,

which is subsequently sampled to produce a new population of better individuals. A positive aspect of EDAs is that the fusion of prior information into the optimization procedure is straightforward, thereby reducing convergence time when such information is available. From a computational cost viewpoint, the amount of heuristics compared with other gradient-free optimization methods is reduced, which means that, in practice, many heuristic optimization methods are not used [48].

For all the above reasons, we selected the so-called Cross Entropy method (CE) [49,50], as the main optimization algorithm for the artificial cognitive control architecture. The most attractive feature of cross entropy is that, for a certain family of instrumental densities, the updating rules can be analytically calculated, making them extremely efficient and fast. Moreover the theoretical background to CE enables theoretical studies of the method, which can provide sound guidelines on the applicability of this algorithm in artificial cognitive architectures.

3.1. Learning

Q-learning is one of the most intensively used reinforcement learning techniques, frequently used to find an optimal policy for Markov decision processes. The problem model, the Markov Decision Problem, consists of an agent, a number of S states and a set of actions per state A. By performing an action $\in A$, the agent can move from one state to another. Executing an action in a specific state provides the agent with a reward. The goal of the agent is to maximize its total reward. It does this by learning the best action for each state. Therefore the algorithm has a function which calculates the *Quality* of a state-action combination, $Q : S \times A \rightarrow \mathbb{R}$

Before learning has started, Q can return any fixed value, chosen by the designer of the problem. Then, each time the agent selects an action, it receives its rewards and enters the new state. The core of the algorithm is a simple value iteration update. It assumes the old value and makes a correction based on the new information:

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \boldsymbol{\alpha_t}(s_t, a_t)(R_{t+1} + \gamma \max_{a \in A} Q_t(s_{t+1}, a) - Q_t(s_t, a_t)) \quad (1)$$

where s_t is the state in time t; a_t is the action taken in time t; R_{t+1} is the reward received after performing action a_t is the learning rate and γ is the discount factor which trades off the importance of sooner versus later rewards.

Normally, Q-learning is executed in an episodic manner where an episode ends when state s_{t+1} is a final state. However, Q-learning can also learn in non-episodic tasks. It may be noted that Q-learning does not specify a method to select the action to perform in each state. However there are several policies to select an action, i.e., the well-known $\in$-*greedy* or *softmax* policies.

In the cognitive architecture, the state is a set of parameters of the model/models, thus each state is identified unequivocally with a set of parameters:

$$s_t \leftrightarrow (K_1^{(t)}, K_2^{(t)}, \ldots, K_N^{(t)}) \quad (2)$$

So, the actions to change from one state to another are those that change at least one parameter of the set $(K_1^{(t)}, K_2^{(t)}, \ldots, K_N^{(t)})$. Thus, the Q-values function is $Q(s_t, a_t) = Q(s_t)$.

The continuous space of the variables is discretized for simplicity as already reported in [51]. So, each parameter K_i has its own limits $[K_i^{min}, K_i^{max}]$ determined by the model to which the parameter belongs. Then, if there are M possible values of each parameter, the range of values of this parameter will be:

$$K_{i_1} = K_i^{\min}, \quad K_{i_2} = K_{i_1} + \frac{K_i^{\min} - K_i^{\max}}{M-1}, \ldots, K_{i_M} = K_i^{\max} \quad (3)$$

As a consequence of the above function, the space of states with a dimension of M^N is finite. Due to the restrictions of a real environment, we cannot take long steps in a specific parameter at a given moment of time. Thus our actions will be limited to comply with this restriction.

For a given state $s_t \leftrightarrow (K_1^{(t)}, K_2^{(t)}, \ldots, K_N^{(t)})$, its available actions will be those that change s_t to $s_{t+1} \leftrightarrow (K_1^{(t+1)}, K_2^{(t+1)}, \ldots, K_N^{(t+1)})$, where:

$$K_i^{(t+1)} \in [\max(K_i^{\min}, K_i^{(t)} + step), \ \min(K_i^{\max}, K_i^{(t)} + step)] \quad (4)$$

In cognitive architectures, as in any hierarchical approach there are different time scales (bandwidths). The learning procedure runs at a lower frequency than the control mechanism, which resembles a cascade concept, because the process has to run for a sufficient length of time, in order for correct learning to take place. Taking this factor into account, if the control mechanism has a sampling time to control of $p_{control}$, the learning has to be performed at least ten times slower than the control, i.e., $p_{learning} = \delta p_{control}$, where $\delta \in \mathbb{N}$, $\delta \geq 10$. The reward function is defined as:

$$R = \begin{cases} +500 & if \ \phi(t) \leq 0.05 \\ +100 & if \ 0.05 \leq \phi(t) \leq 0.1 \\ -100 \cdot \phi(t) & if \ 0.1 \leq \phi(t) \end{cases} \quad (5)$$

where the performance index associated with the action that is taken, $\phi(t)$, has the following expression:

$$\phi(t) = \sum_{i=1}^{\delta} \left(\frac{ref_i - y_i^{(t)}}{ref_i} \right)^2 \quad (6)$$

in which, ϕ_t is the reference value in time $t + i \cdot p_{control}$ and $y_i^{(t)}$ is the output of the process in time $t + i \cdot p_{control}$ with the parameter set $(K_1^{(t)}, K_2^{(t)}, \ldots, K_N^{(t)})$. As we can see, $\phi(t)$ is the mean square error evaluated in $[t, t + i \cdot p_{control}]$.

Taking into account these modifications, the function to update the *Q-values* is:

$$Q_{t+1}(s_{t+1}) = Q_t(s_{t+1}) + \alpha_t(R_{t+1} + \gamma \max_{a \in A} Q_t(s_{t+2}) - Q_t(s_{t+1})) \quad (7)$$

For the sake of simplicity, we used the $\in$-*greedy* policy in this first approach, to choose an action, because it is sufficient in almost all scenarios. The $\in$-*greedy* policy algorithm is shown in Fig. 2.All the steps in the modified Q-learning algorithm are presented in Fig. 3.

3.2. Self-optimization capability

Let X be a random variable defined in the space χ and $f : \chi \rightarrow \mathbb{R}$ a score function. The CE method seeks to find x' such that

$$\gamma' = f(x') = \min_{x \in \chi} f(x) \quad (8)$$

The algorithm transforms this problem into an associated stochastic problem by defining a family of random variables with density function $g(x, v)$, $v \in \Gamma$ and solving it as the simulation of a rare event, where the event is sampling around the optimum of f. The algorithm can be summarized as follows:

1. Initialize v_0.
2. Generate a sample of size N, $(x_i^t)_{1 \leq i \leq N}$, from the density function $g(x, v_t)$. Let $f_1 \geq f_2 \geq \cdots \geq f_N$, $f_i \in \{f(x_i^t)\}$, $1 \leq i \leq N$ be the corresponding ordered score values and $\gamma_t = f_{\lfloor \rho N \rfloor}$.
3. Update v_t to

$$v_{t+1} = \operatorname*{argmin}_{v} \frac{1}{N} \sum_{i=1}^{N} I_{\{y \leq \gamma_t\}}(f(x_i^t)) \cdot \ln(x_i^t, v_t) \quad (9)$$

4. Repeat from step 2 until convergence or ending criterion.

Algorithm 1: ∈-greedy police algorithm

```
r = ramdom()
if r < ∈
    Take a random action between all possible actions.
else
    Take the action that produces the state with most Q-values.
end
```

Fig. 2. Algorithm for ∈-greedy policy.

Algorithm 2: Modified Q-learning algorithm

```
Initialize Q(s_t) arbitrary (or with a fixed value obtained with some method)
Initialize s_0 to an arbitrary or fixed state
repeat
    foreach step do
        Choose a_t using the ∈-greedy police algorithm;
        Perform action a_t and change to s_{t+1};
        Wait δ•p_control and recieve R;
        Update Q-values with equation (7);
        s_t ← s_{t+1};
    end
until s_t is a terminal state;
```

Fig. 3. Modified Q-learning algorithm.

5. Assuming that convergence has been reached at $t = t'$, the random variable defined by the density function $g(x, v_{t'})$ should have all of its mass concentrated on x'.

Step 3 is performed using the best $[\rho N]$ samples, also called elite samples. The sampling density function needed in the 2nd step is usually unknown, but in most cases it can be assumed to be a normal distribution function. In this case, v represents the mean μ and the standard deviation of the normal distribution is σ. The solution of the equation is simply the sample mean $\bar{\mu}_t$ and sample deviation $\bar{\sigma}_t$ of the elite samples. It also follows that the mean should converge to x' and the deviation should converge to zero. A smoothing parameter α for the mean vector and dynamic smoothing β_t for the deviation are applied, in order to prevent the occurrences of 0 s and 1 s in the parameter vectors.

$$\begin{aligned} \bar{\mu}_{t+1} &= \alpha\bar{\mu}_{t+1} + (1-\alpha)\bar{\mu}_t \\ \bar{\sigma}_{t+1} &= \beta_t\bar{\sigma}_{t+1} + (1-\beta_t)\bar{\sigma}_t \\ \beta_t &= \beta - \beta\left(1 - \frac{1}{t}\right)^q \end{aligned} \tag{10}$$

where $0.4 \leq \alpha \leq 0.9$, $0.6 \leq \beta \leq 0.9$, $2 \leq q \leq 7$.

Finally, constrained optimization problems will be addressed from an engineering viewpoint, which therefore means imposing boundaries on the distribution function for the generation of samples, to ensure that sampling is from within the appropriate region.

4. Design and implementation of the artificial cognitive architecture

In this section, we will present the design of the artificial cognitive architecture together with the requirement analysis. We will then explain the main details of our implementation without entering into deep explanations. Having presented our artificial cognitive architecture in detail we then present an instantiation of the architecture to validate its design and implementation.

4.1. Design and implementation

The artificial cognitive architecture should comply with both functional (FR) and non-functional (NFR) requirements. The main functional requirements can be summarized as follows.

FR1 **Control architecture**: the main function of this architecture is to control processes; the implementation of the architecture must allow the user to assign a process to the architecture and prepare the architecture to control it.

FR2 **Models**: the architecture has several models that serve to control a process with different procedures. There are four types of models: *single loop* models, *direct* and *inverse* models, and *simulation* models. The configuration of *direct* and *inverse* models resembles the internal model control paradigm well-known in the Control Engineering community, but also claimed as the main explanation and rationale behind brain-cerebellum interaction [16].

FR3 **Modes**: the architecture must run in different modes. A mode is defined by a preset configuration of the different elements of the architecture (models, reference values and process entity) to control a process. A mechanism enables the application to switch between modes while running. Switching can be smoothed out by a first-order filter to guarantee a seamless transition from one mode to another one.

FR4 **Adaptation**: the application must provide a component for selecting models required by a specific mode. The choice of component may be to accomplish different objectives.

FR5 **Optimization**: the architecture must provide functionality for optimal setting of control models on the basis of a *simulation* model of the physical process. With this action, the architecture will be able to improve its behavior while running different processes.

FR6 **Online learning**: similar to optimization, the architecture should provide a mechanism to execute a learning algorithm during the regulation process. Once again, this mechanism will improve the behavior of the overall system.

FR7 **Objectives**: the architecture must ensure that the user inserts the objectives to be achieved, e.g., *productivity*, *performance*, etc.

FR8 **Data types**: the architecture must allow different data types, such as integer, double or string.

The main non-functional requirements are described as follows.

NFR1 **Middleware**: the architecture shall be quite generic and flexible to allow the user to use it over a middleware, for instance, the user may wish to use the architecture to control a process in a different place, i.e., to distribute the architecture to control remote process.

NFR2 **Extensibility**: the architecture shall be designed to ease the tasks of adding models, control algorithms, optimization and learning mechanisms, etc.

In order to comply with the above-mentioned requirements, we have designed an object-oriented library. Along with the general classes and interfaces, some classes are provided to ease the instantiation tasks of the architecture. The following will briefly explain the most important concepts of this library, contained in the *COGNETCON* packages with different functions http://gamhe.eu/downloads/?route=.%2FCognetcon. So as to assess the main

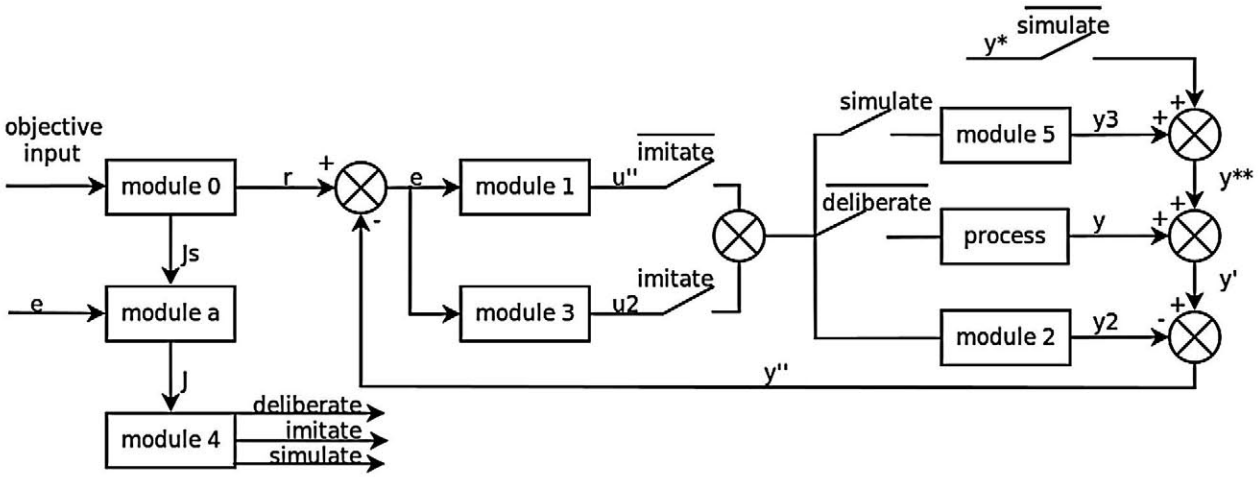

Fig. 4. Overall scheme of the modified shared circuits model (MSCM) [3].

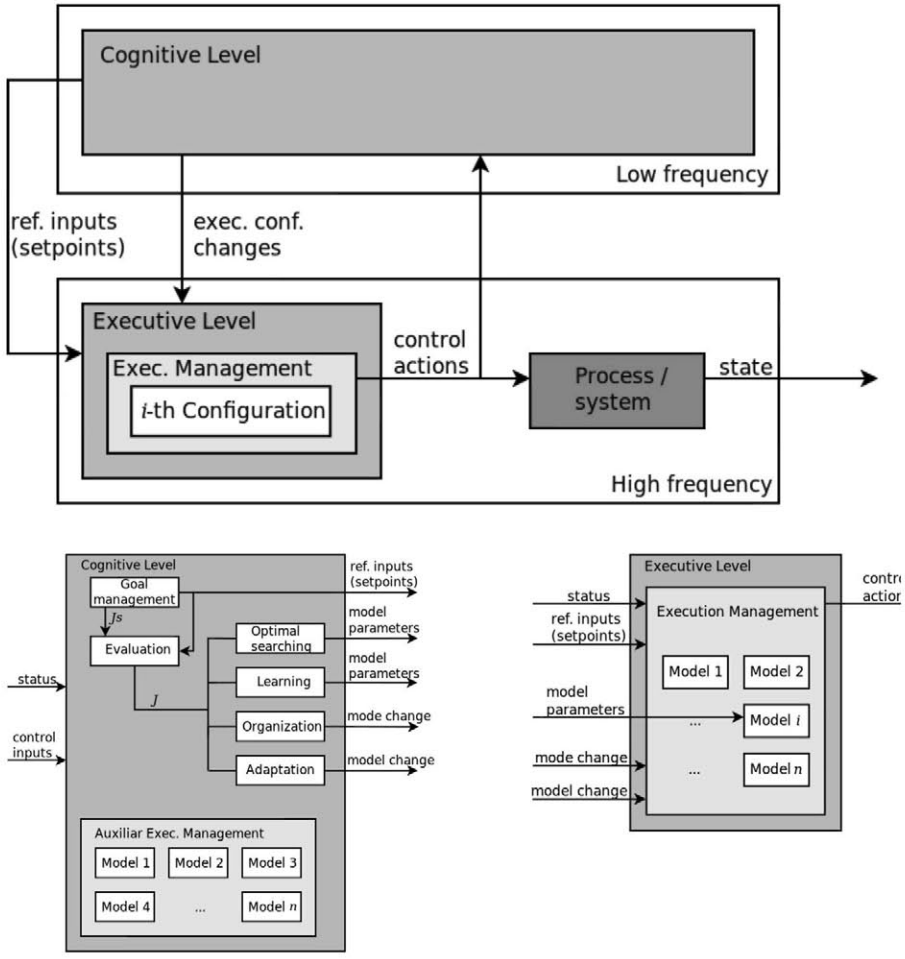

Fig. 5. Overall diagram of the artificial cognitive architecture and the corresponding *cognitive* and *executive* levels.

differences in the implementation of the artificial cognitive control architecture the overall scheme of the MSCM architecture [3] and the scheme of the new design and implementation of the artificial cognitive control architecture are shown in Fig. 4 and Fig. 5, respectively.

A detailed description of the *cognitive* and the *executive* levels is shown in Fig. 5 to gain a better understanding of the interconnection between the different parts of the architecture. There is a mirror of the *execution* level at the *cognitive* level, necessary for *organization*, *adaptation*, and *learning* mechanisms that make use of real-time simulations. Otherwise these mechanisms would have to use the processing time of the *execution* unit, thereby limiting real-time performance.

On the other hand, in the *executive* unit, the most important input is the *exec.conf.changes* that serve to introduce modifications both in configuration and parameters. Parameter modifications change the parameters of a particular model (model i) from the learning mechanism and changes to its configuration can be of two types: new assignment of a model to a specific mode or switching between modes.

Therefore, the new implementation of the architecture is driven by the concept of the model, at the lowest level, and the concept of type of models. Module 1 corresponds to a *single loop* model (i.e., simple feedback control loop), Module 2 matches with *direct* type model (i.e., direct or forward dynamic of the system), and Module 3 is identified with *inverse* type model (i.e., inverse dynamic of the system). Therefore, the architecture is designed so that the user can easily add more types of models, which can lead also to other types of functioning modes. Now, it is necessary to address the concept of executing modes that is the mechanism for defining how to connect the different types of models for a certain operating mode of architecture. For the sake of clarity, only three operating modes are considered which are single feedback control, inverse control and internal model control with the novelty of incorporating self-optimization and self-learning. Similarly, Haykin et al. [52] have also considered three main operating modes. The implementation of the above architecture is coded in Java, which guarantees greater portability between different operating systems.

Note that the type of model is only described in the operating modes and not the model itself. The *adaptation* is in charge of choosing the type of model for each mode depending on the main process characteristics, know-how and available models. The *organization* mechanism is responsible for switching between execution modes without deciding on the type of model used for each mode. Therefore *adaptation* is responsible for the transition between running modes and *Execution Configuration*, which is the concept that permits the interconnection of models in *Execution Management*. *Learning* is performed in MSCM through Module 2, while here it is done by a specific component that enables learning in all models of the architecture. A further component of the architecture, *Optimal Searching*, is in charge of optimization on the basis of *simulation*-type models (e.g., any computational representation of the controlled system that resembles well-known output-error models). Therefore, the ability to emulate Module 5 can be carried out by *simulation*-type models.

The organization procedure is essential to enable the artificial cognitive control to change from one execution mode (e.g., single loop) to another one (e.g., inverse model for inverse control). The overall scheme for organization is shown in Fig. 6. For the sake of clarity, *inverse* model enables the inverse control mode that corresponds to anticipation and mirroring for priming, emulation and imitation from the viewpoint of neuroscience. The combination of *direct* and *inverse* models enables the internal model control that corresponds to the mindreading effect and the roots for emulating socio-cognitive skills from the perspective of neuroscience.

The initialization procedure starts with the optimal setting of parameters of each model (inverse model, direct model, single loop model) on the basis of the cross entropy method introduced in Section 3.1 and the simulation model of the process. Indeed, self-optimization is a basic step that can serve to carry out other tasks beyond this one. The use of an error-based performance index and a rough model of the process is enough to perform this task as it is shown later in the proof of concept of the artificial cognitive control system in Section 5.2.

Moreover, necessary variables for learning and organization will be initialized such as the Q-learning tables and the performance index table, in order to be later applied to set the initial conditions in the organization as well as to choose the appropriate execution mode. The performance index table is also updated using a forgetting factor.

The self-organization procedure during each execution cycle consists of eight main steps which are also depicted in Fig. 6. First, the system checks if the control signal (action) has yielded the expected results according to the performance index, if so, the artificial cognitive control remains in this execution mode (e.g., single loop control). If the execution mode does not achieve

Algorithm 3: Self-organization algorithm

```
1  if Is the last action successful?
2  |   Maintain the last mode
3  else
4  |   if Is learning active for that mode?
5  |   |   if Has it been learning for enough time?
6  |   |   |   Update the performance indices table
7  |   |   |   Obtain the possible actions
8  |   |   |   Choose the best action
9  |   |   else
10 |   |   |   Wait for another cycle
11 |   |   end
12 |   else
13 |   |   activate learning for that mode
14 |   end
15 end
```

Fig. 6. Organization procedure for enabling the artificial cognitive control system.

appropriate results it is necessary to verify if the learning is enabled, otherwise, it is activated. The duration of the learning (set by the user) should be checked in the third step. The performance index table is updated with the real-time computed value with a forgetting factor. All possible control signals (actions) are then computed and the control signal that produces the best behavior (i.e., the best performance index) is taken from the table.

4.2. Instantiation and deployment

Both the design and the implementation of the architecture require an instantiation for its validation. We will try to replicate the configuration, making an instantiation with three modes of operation that control a micro-drilling process. Certain specific assumptions are implicit in this instantiation, so that that the artificial cognitive architecture can be deployed in low-cost computing platforms. Under this rationale, an analysis of the best available middleware to enable networked, transparent, portable and reliable communication, and low-cost computing platforms is necessary to select the best one for the instantiation.

4.2.1. Middleware

The scientific community is currently at work to connect and to integrate sensors with other devices that will improve factory production. One key issue in the endeavor is the long-distance monitoring and control of complex plants, which requires synergetic strategies that link smart devices and communication technologies with advanced computational methods [53]. The solution chosen for this work employs distributed object-computing middleware, which enables common network programming tasks to be automated, regardless of other considerations such as what communication protocols and networks are used to interconnect the distributed objects.

The first middleware option analyzed was Common Object Request Broker Architecture (CORBA). CORBA technology provides a clear opportunity for process monitoring and strategic process control. In real-time CORBA (RT-CORBA) specification, mechanisms and policies are defined to control processor resources, communication resources and memory resources to support the real-time distributed requirements of the application fields [54]. The second option was ZeroC Ice that provides a simple

and easy-to-understand communication solution. Yet, despite its simplicity, Ice is flexible enough to accommodate even the most demanding and mission-critical applications. A comparison with other popular distributed computing solutions can be found here [55]. One of the most important features of Ice is its enhanced set of services, such as event distribution, firewall transversal with authentication and filtering, automatic persistence, automatic application deployment and monitoring, and automatic software distribution and patching. All services can be replicated for fault tolerance, so as to avoid the introduction of any single point of failure. The use of these services greatly reduces development time, because they eliminate the need to create distribution infrastructure as part of the application development. The third solution that was explored, Java Remote Method Invocation (RMI), enables the programmer to create distributed Java technology-based applications, in which the methods of remote Java objects can be invoked from other Java virtual machines, possibly on different hosts. RMI uses object serialization to marshal and unmarshal parameters and does not truncate types, supporting true object-oriented polymorphism.

Although the design and application of artificial cognitive control architecture based on middleware is essential, because the middleware facilitates communication between different hosts, the design of the architecture and its development is done in a middleware-free manner; independent of the middleware chosen to enable communication. ZeroC Ice was selected for the implementation, on the basis of good results previously reported in the literature, its advantages in relation to CORBA, its versatility and ease of use [56]. Three distributed units were considered in the deployment of the artificial cognitive control architecture, on the basis of the design and specificities of the case study with regard to proprietary software.

Cognitive unit: This unit contains the components of the *cognitive* level: learning and optimization mechanisms, organization logic, execution logic and the application itself. It is expected that this unit will be deployed in a low-cost computational hardware.

Executive unit: This unit contains the models that are used in the *single loop*, *direct*, *inverse* and *simulation* modes of the architecture. Once again, the unit is expected to be deployed in a low-cost computational hardware.

Process unit: A distributed process is needed, because that is one of the objectives, i.e., the control of a physical process.

The Ice specification file is programmed with this description and parses it with the *slice2java* program. This program will generate several auxiliary classes needed by ZeroC Ice for communication over the net. In addition to the common use of Ice, *IceGrid*, is an important service that enables clients to discover the corresponding servers. Acting in an intermediary role, *IceGrid* decouples clients from their servers and is intended to improve the performance and reliability of applications through support for replication, load balancing and automatic fail over. The program needed to execute the IceGrid registry process, *icegridregistry*, and the program to run a server, *icegridnode* are both provided with the Ice installation package.

4.2.2. Low-cost computing platforms

Various state-of-the-art low-cost computing alternatives were analyzed, to choose the most suitable one to meet our objectives for deployment in a low-cost computational platform with artificial cognitive control. We reviewed three of the most popular low-cost computing platforms reported in the literature: Raspberry Pi Model B, HummingBoard-i2 and BeagleBone Black. These computing platforms also have forums which share posts from the community of users that help others to make efficient use of the hardware.

Raspberry Pi is a low cost, credit-card sized computer capable of everything expected from a desktop computer, from browsing the Internet and playing high-definition videos, to the use of spreadsheets, word-processing, and gaming [57]. It is a small versatile low-cost device with a 700 MHz Single-Core ARM v6 and 512 MB SDRAM that enables users of all ages to explore computing, and to learn programming languages such as Scratch and Python. Raspberry Pi can interact with the environment and devices in a wide range of digital maker projects, from gaming machines [58] to open-source voice computing [59].

The HummingBoard-i2 and Raspberry Pi both share a very similar layout and configuration, making transition projects between both of them very easy. The former represents a good choice, if users are looking for a more powerful option with a 1.0 GHz Dual-Core ARM v7 and 1GB SDRAM at over twice the cost of Raspberry PI.

BeagleBone Black is suitable for users looking for a little more power that Raspberry Pi, an easier set up, easier commercialization, or users who have a need to interface with many external sensors. Its configuration consists of an AM335X 1 GHz ARM Cortex-A8 and 512 MB of DDR3 RAM at a similar cost to Raspberry PI.

World-wide support for a low-cost computing platform to facilitate implementation and to solve deployment problems and, less importantly, the cost of the platform, drove our decision-making process. A large user community provides ample support for trouble shooting when using the platform. We finally chose Raspberry Pi Model B, partly because it is the most popular platform with an active user community.

4.3. Tools for modeling and implementation

UML is a general-purpose modeling language in the field of software engineering, which is designed to provide a standard means of visualizing the system design. It was applied to re-designed and to implement the artificial cognitive architecture [60]. Java was selected as the programming language, because of its universal portability in different environments. In addition, the Real Time Specification for Java (RTSJ) provides an advantage when we want to extend our work to a full real time environment [61].

Having completed the UML-based design, the next step is the implementation of the architecture was the selection of *Eclipse*, as the integrated development environment (IDE) that facilitates work with Java. We also used SWIG [62], an interface compiler that connects programs written in C and C++ with several languages such as Perl, Python or Java. SWIG permits the re-use of models programmed in C/C++ and performs its tasks through the Java Native Interface (JNI) framework that enables assembly and communication between the Java Virtual Machine (JVM) and programs written in C, C++.

5. Real application experiment

The setup of the real application experiment to validate the artificial cognitive architecture is presented in the following section. The main results obtained after running real-time experiments are also discussed.

5.1. Experiment setup

Due to the small dimensions involved in the microdrilling processes, the control is very difficult to carry out online. For this kind of processes, the use of online indirect monitoring is particularly important. Measurable process signals such as forces, vibration, acoustic emission and motor current have been often used for this purpose. Online quality control systems can provide real-time information, which can be supplied as a feedback to CNC

Table 1
Equivalence between the MSCM modules [3] and the proposed architecture.

Modified SCM modules [3]	Proposed architecture
Module 0. Objective management	Goal management
Module A. Performance index computation	Performance indices computation
Module 1. Basic adaptive feedback control	Modes and models (single loop)
Module 2. Simulation prediction of effects for improved control	Modes and models (direct or forward models)
Module 3. Mirroring for priming, emulation and imitation	Modes and models (inverse models)
Module 4. Management of monitored output inhibition	Organization
Module 5. Counterfactual input simulation	Operating modes and models
Not available	Adaptation
Not available	Optimal searching/optimization (simulation models)
Not available	Learning (simulation models)

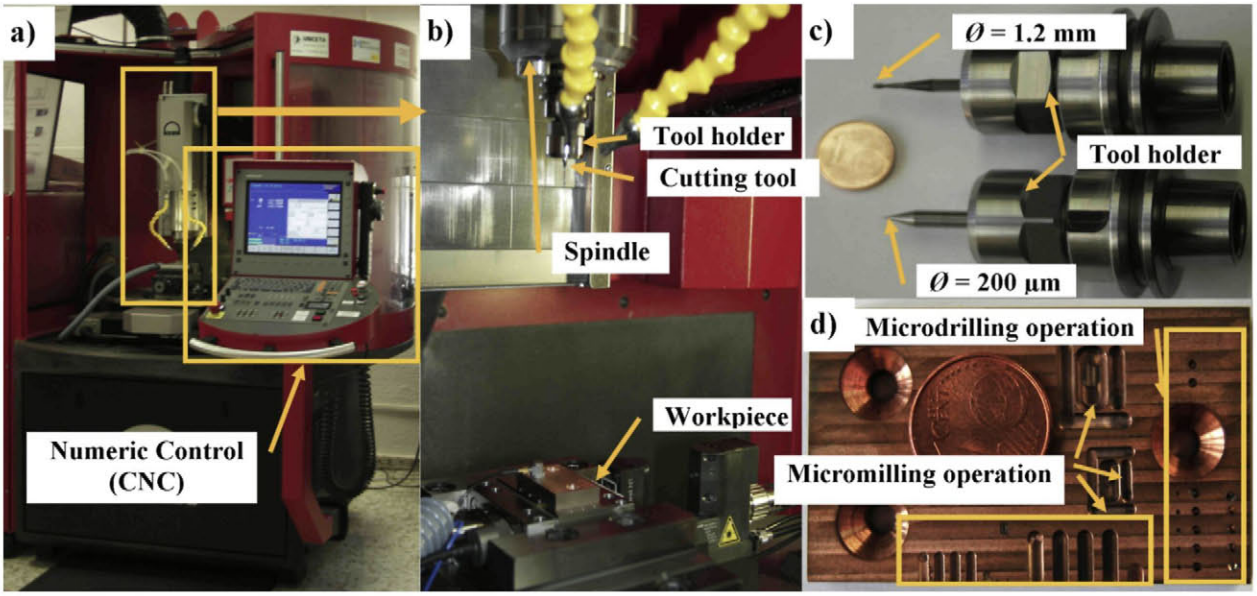

Fig. 7. Setup of the industrial environment for microdrilling processes.

for online adjusting cutting parameters. These kind of systems has been proposed for conventional machining. Monitoring and control systems have been also proposed in order to adaptively modify the cutting parameters in real-time for guaranteeing the geometric quality [63].

In the experimental study, the instantiation was tested in an industrial environment. All cutting operations were done on a Kern Evo Ultra-Precision Machine Centre (see Table 1), equipped with a Heidenhain iTNC540 CNC. Maximum spindle speed (n) and feed rate (f) were 50,000 rpm and 16,000 mm/min, respectively. The experimental platform included a cutting force sensor on three axes, two vibration sensors for y, z axes and a laser sensor for measuring variations in tool length and radius. The measurement of cutting force signals was done with a multi-component dynamometer. A Kistler sensor (MiniDyn 9256C1) was used on the z-axis, with a bandwidth of up to 5 kHz. The vibration signals were measured by two accelerometers attached with wax to both the y and the z-axes of the work piece. Tool length was measured on-line with a laser sensor. This high precision visible red-light laser is a state-of-the-art measurement system, fully up to date and commercially available. All the sensor signals were fed into a NI 6251 National Instruments data-acquisition card, with a sampling rate of 50 kHz, and were processed with a National Instruments high-performance PXI-8187 embedded controller. The position of the tool tip (x, y, z) was obtained via the ethernet connection of the CNC of the machine, using a sampling frequency every 12 ms (83.33 Hz). Fig. 7 shows the platform with its main devices that are labeled.

Raspberry Pi (*Pi 1*) and Rasberry Pi (*Pi 2*) run the cognitive and executive part of the architecture, respectively. *Process host* has the mission of retrieving the process outputs from the *KERN Evo* machine and sending them to the architecture via ZeroC Ice, as well as receiving the action control from the architecture and sending it to the *KERN Evo* machine. As shown in the picture, communication between *Process host* and *KERN Evo* is done over the *Ethernet*. The *Icegrid registry* program is permanently running on the *Registry* host to enable the different hosts to identify each other. Finally, the *Client* host can use the developed graphical user interface to interact with the different components. The deployment of all components in the architecture is shown in Fig. 8(a) and (b).

5.2. Results

We controlled microdrilling process force in this experimental setup. Several experiments were performed to validate the design and implementation of the artificial cognitive architecture. The experiment consisted of single and consecutive (10 holes) 0.5 mm drilling operations at a spindle speed of 10,000 rpm and a feedrate of 100 mm/min. First, in order to assess each control mode (i.e., single loop, inverse control and internal model control) only one drilling action was carried out.

The control strategies summarized in Table 2 were selected to perform the proof of concept of the artificial cognitive control. It does not mean that other control strategies cannot be selected. The user can define and configure models and control strategies in the artificial cognitive control architecture. The main rationale for using fuzzy and neuro-fuzzy approaches can be briefly summarized as follows.

Neuro-fuzzy systems combine their ability to accurately model any nonlinear function, an excellent learning capacity, and an ability to represent human thought and robustness in the presence of noise and process uncertainty. These characteristics are essential to deal with uncertainties, nonlinearities, and time-varying behavior. Some neuro-fuzzy systems cannot be applied to real-time process control, mainly because of the computational cost involved and because they are unable to meet the control system's requirements.

One hybrid strategy that has been successfully applied is the ANFIS system, due to its computational simplicity and suitability for real-time applications [64]. A complete review of neuro-fuzzy and new fuzzy inference systems [65] goes beyond the scope of this paper, although a thorough study should be carried out in a near future. The use of a fuzzy control system for the single loop is fully justified on the basis of the nature of micromanufacturing processes. The last decades have witnessed plenty of successful industrial applications of fuzzy controllers as well as dozens of

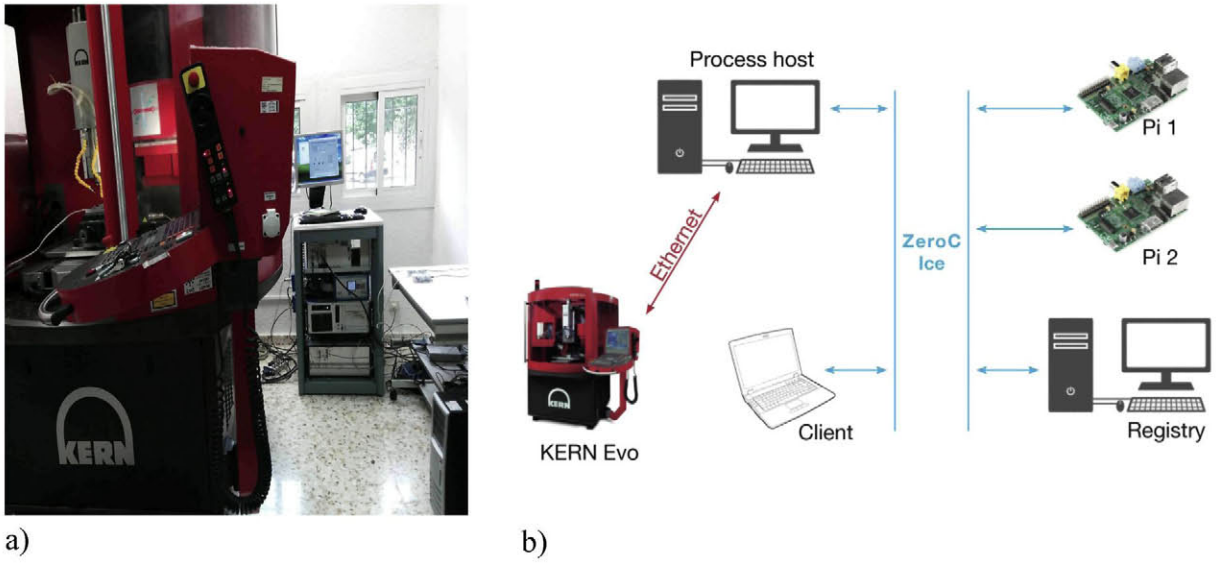

Fig. 8. (a) Overall view of the industrial setup, (b) schematic diagram of the architecture of artificial cognitive control.

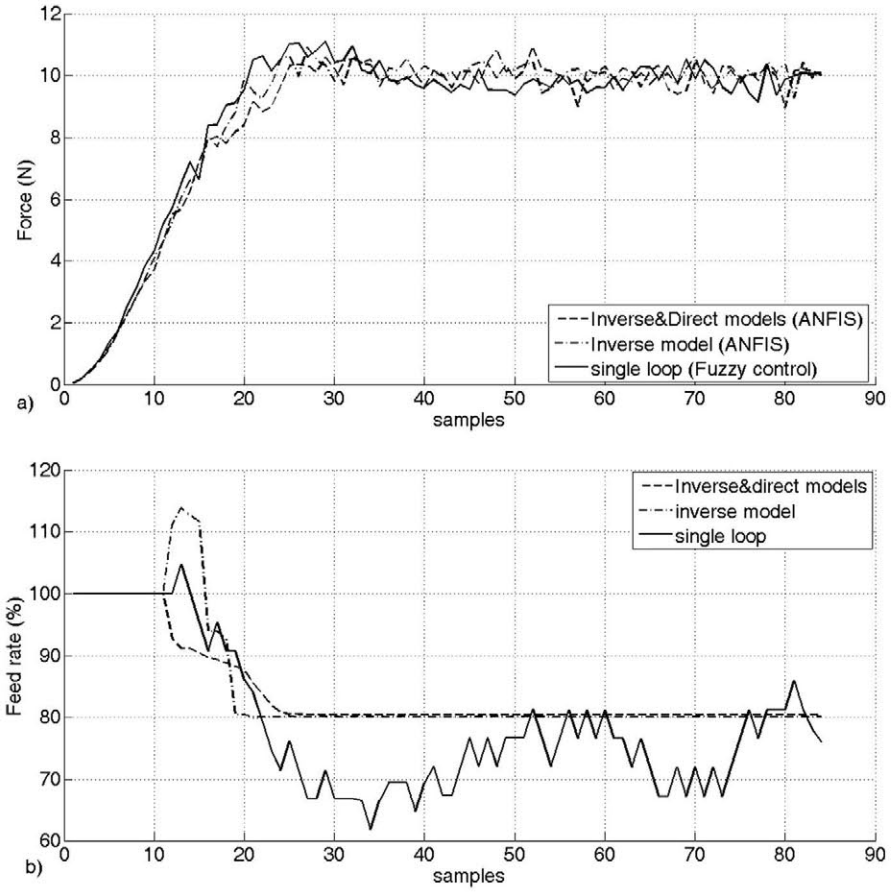

Fig. 9. Behavior of the drilling force (a) and the control signal (feed rate) (b) for each execution mode isolated on the basis of the simulation.

Table 2
Algorithms implemented for each execution mode.

Implemented algorithms	Single loop	Direct model	Inverse model
Algorithm	Fuzzy Logic	ANFIS	ANFIS
System	Two-input/single output	Single-input/single output	Single-input/single output
Inputs	Error (e) and change in Error (de)	Feed rate (f)	Force (F)
Outputs	Feed rate (f)	Force (F)	Feed rate (f)
Membership functions type	Triangular-shaped	Gaussian	Gaussian
Number of membership functions	7	3	3
Inference system	Mamdani	Takagi-Sugeno	Takagi-Sugeno
Number of rules	49	9	9
Defuzzification	Center of area	Weighted average	Weighted average
Iterations	–	100	100
Learning rate	–	0.01	0.01
Training algorithms	–	GENFIS 2	GENFIS 2
Training data set	–	62 samples	62 samples
Validation data set	–	82 samples	82 samples

books and surveys on this topic. New research trends and industrial application of fuzzy control are well addressed in [66].

Fig. 9 depicts the results of the simulation of each control mode. In order to carry out this simulation the model of the microdilling process represented in Eq. (11) is considered as well the influence of noise represented by (12). Based on the technical knowledge of the process an input and output system is considered using the following variables: input is the feed rate (f) and output, the cutting force (F). For the study an approximate representation of the process behavior was used. The linear model represented via difference equation is expressed accordingly:

$$F(k) = a_1 \cdot f(k) + a_2 \cdot f(k-1) + a_3 \cdot f(k-2) + a_4 \cdot f(k-3) - b_1 \cdot F(k-1) + b_2 \cdot F(k-2) + b_3 \cdot F(k-3) \quad (11)$$

where $f(k)$ is the feed rate and $F(k)$ is the cutting force at k-instant. The coefficients of the difference equation are $a_1 = 0.004322$, $a_2 = 0.02467$, $a_3 = 0.008623$, $a_4 = 0.00002178$, $b_1 = -2.447$, $b_2 = 1.993$ and $b_3 = -0.5406$.

This model roughly describes the dynamic behavior of the drilling process and it has been verified experimentally. However, model parameters depend on the workpiece material, cutting conditions, and tool wear. Therefore, model coefficients are time variant and variables of cutting conditions, workpiece material and tool wear. A disturbance $d(t)$ (12) or noise input is included, in order to better replicate a real industrial process:

$$d(t) = A \cdot (\sin(2\omega t) + \sin(3\omega t) + \sin(4\omega t) + \sin(5\omega t)) \quad (12)$$

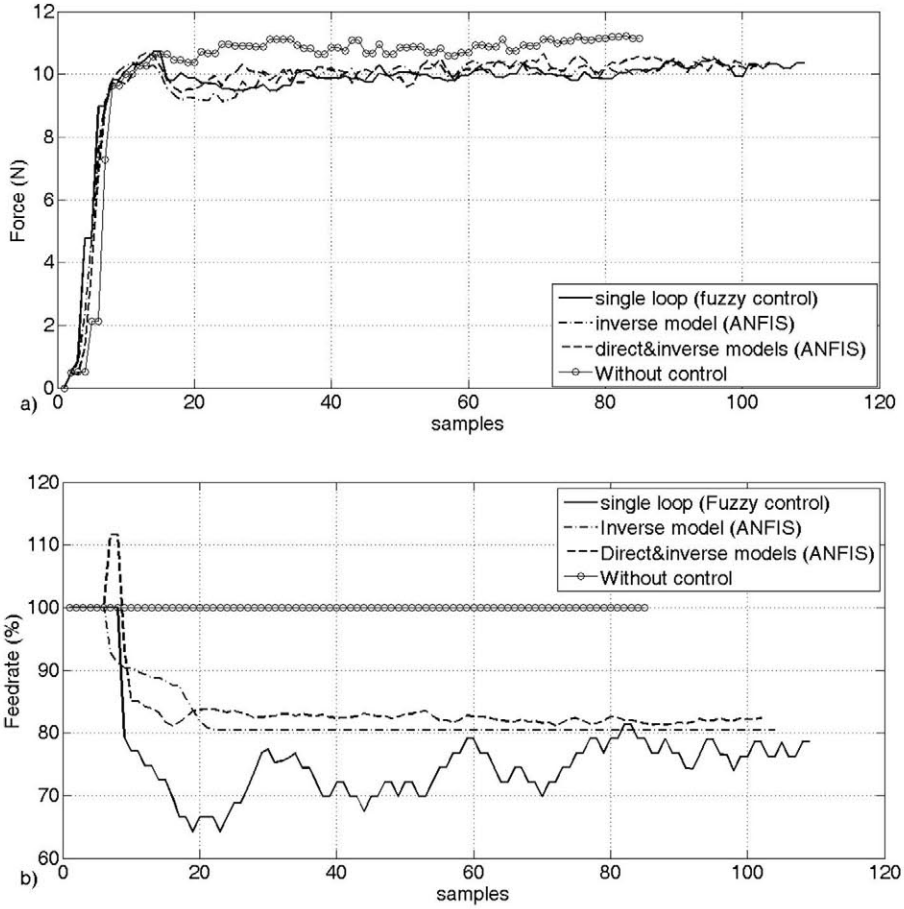

Fig. 10. Behavior of the drilling force (a) and the control signal (feed rate) (b) for each execution mode isolated from real time experiments.

Table 3
Performance indices for real time experiments.

Controller	*ISE*	*AAE*	*MSE*	*Ovt* (%)
CNC alone (without control)	559.79	1207.39	25.66	12.15
Single loop (fuzzy control)	338.45	628.23	17.62	7.16
Inverse model (inverse control)	380.40	706.76	19.13	5.71
Inverse and direct models (IMC control)	418.81	742.48	20.26	6.74

where $\omega = 7.61$ rad/s. This frequency corresponds to the greatest frequency of the poles of the third-order system model of the drilling process given in (11). The amplitude of the disturbance is $A = 10$ (about 10% of additive noise).

Fig. 9 shows the behavior of drilling force in real time for *three modes* (single loop, inverse control and internal model control) of operations as well as the behavior of the drilling force without control (i.e., at constant feed rate). Each control mode is running isolated in each microdrilling operation. For the sake of space, the experimental results depicted in Fig. 9 are running after the optimal setting of parameters on the basis of the cross entropy method. The dynamic response for the three operating modes is appropriate after running the initialization of the artificial cognitive control system. The optimal setting of parameters for each control modes is performed using the rough model of the process (11) and the mean square error performance index (Fig. 10).

Table 3 shows a comparative study of the single loop operating mode with the inverse model (i.e., inverse control) and the inverse and direct models (i.e., internal model control). The integral of square error (*ISE*), the average of absolute error (*AAE*), the mean square error (*MSE*) and the overshoot (*OVt*) are used. The error performance indices of the single loop are better than the other controllers whereas the inverse control (inverse model) yields the better overshoot. It is important to remark that the microdrilling process is non-linear and time variant process, and therefore the performance of each control mode may deteriorate due to nonlinearities, uncertainty and time-variant behavior of the process.

Experimental results using the platform shown in Fig. 8 are shown in Figs. 11 and 12.

Fig. 11 shows the behavior of the drilling force when the learning is activated for the inverse model (i.e., inverse control mode). The drilling of 10 holes is performed, in order to show the influence of reinforcement learning on improving the performance of the inverse model. Initially the response is very poor and the system cannot reach the set point. The behavior of the drilling force is quite good from the 7th hole due to reinforcement learning.

Fig. 12 shows the behavior of the drilling process of nine holes. The single loop mode (i.e., the fuzzy control) is functioning in the first three holes. After that, the poor performance index motivates the change to the internal model control where *direct* and *inverse* models are activated. This is a clear case study where the single loop is deteriorating due to the influence of disturbance such as

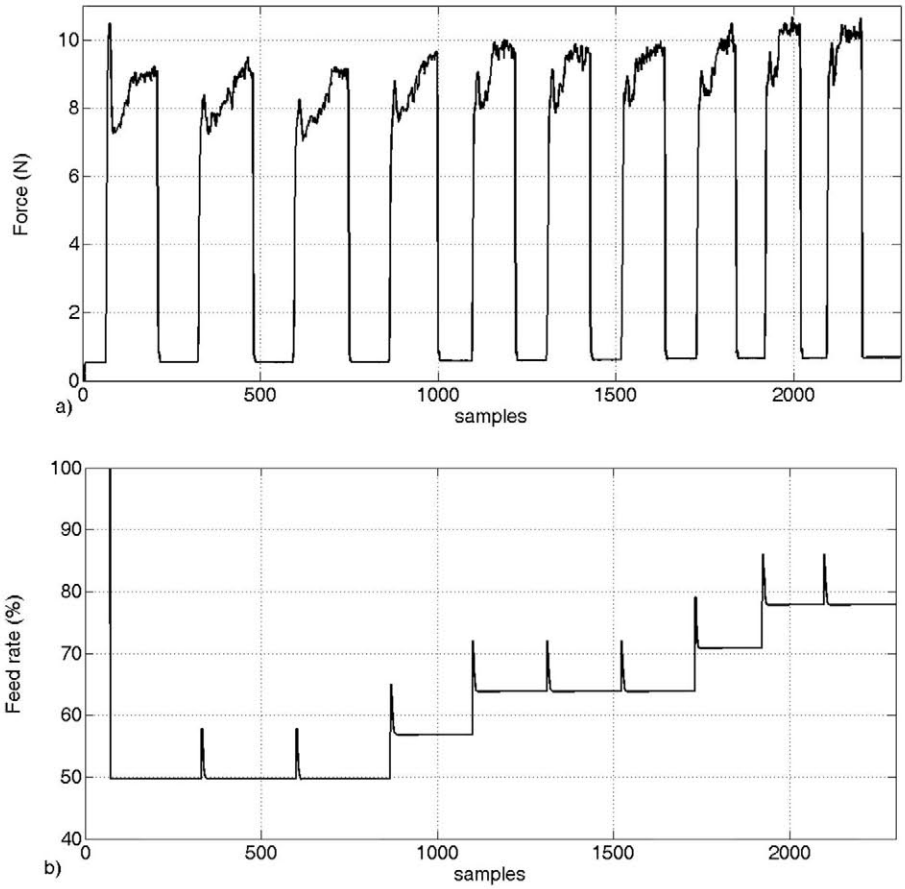

Fig. 11. Reinforcement learning for the inverse model in real-time microdrilling of 10 holes of 0.5 mm diameter, (a) behavior of the drilling force, (b) control signal represented by the feed rate.

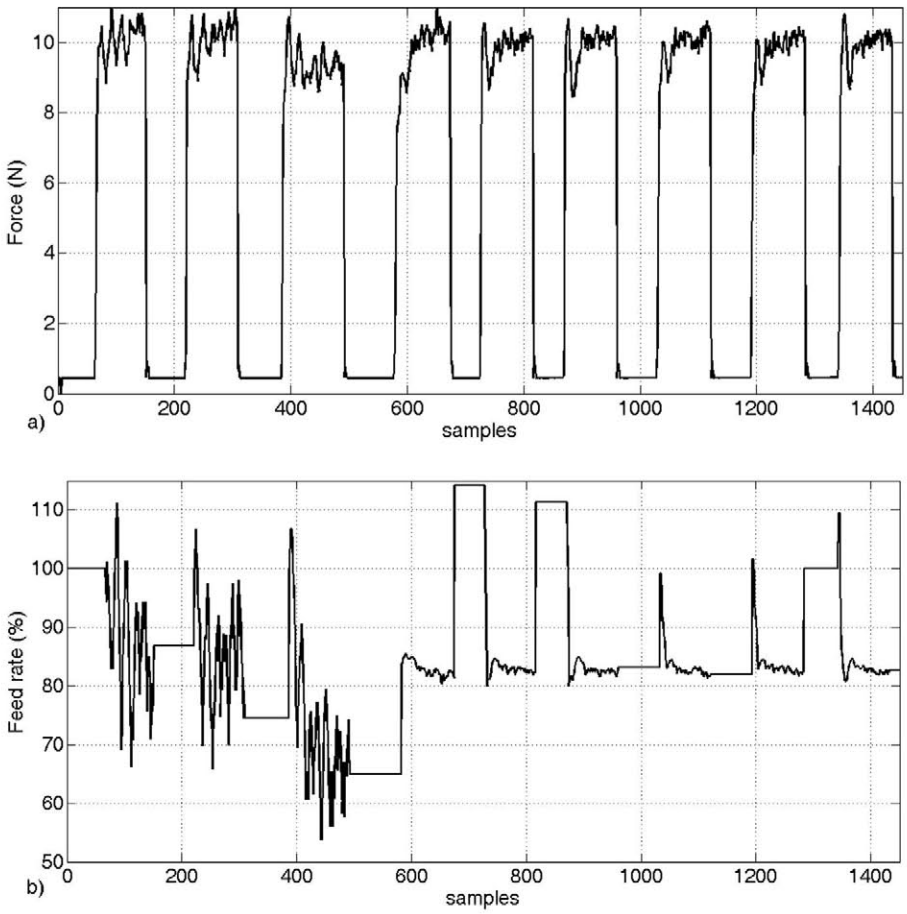

Fig. 12. Behavior of the drilling force (a) and the control signal (b) in the nine holes of 0.5 mm.

tool wear and the artificial cognitive control tries to find an adequate solution by changing the execution mode. The dynamic response and the performance index are then better in the new execution mode and the system remains in this mode. Therefore, the organization strategy depicted in Fig. 6 and described in the Section 4.1 is adequate for this case study.

There are some issues to be analyzed. The offsets in the dynamic response shown in Figs. 11 and 12 are the result of the influence of air for tool refrigeration when the force is measured. This negative effect is not easily removed because it is not a constant value. The second issue is the parametrization of the threshold in the performance index to change from one mode to another one, which depends mainly on the application. The log files of the organization shown in Figs. 11 and 12 are depicted in Annex I.

6. Conclusions and future work

This paper has presented the design and implementation of an artificial cognitive control system in a low-cost computing platform with self-optimization and self-learning capabilities. Firstly, an instantiation of the artificial cognitive architecture has been designed and developed. For the sake of clarity and to provide new capabilities to the architecture, the optimization procedure ran on the cross-entropy algorithm and the online learning mechanism on the Q-learning algorithm, both of which were designed and implemented in the architecture.

In addition, classes were developed that provide this instantiation with the ability to run in a distributed manner. The overall assessment of the instantiation was performed in a simulation study and a real manufacturing environment, both of which yielded very promising results. Beyond the case-study on force control for microdrilling processes and the results that have been presented, the artificial cognitive control architecture built on a low-cost platform hardware has demonstrated the suitability of the implementation in an industrial setup. The functional and non-functional requirements are fully satisfied by means of a simple instantiation configured with middleware.

This research work has provided an important starting point to address the main challenge of an artificial cognitive approach embedded in low-cost hardware industrial computing on the basis of low-cost hardware. Fully aware of the preliminary nature of this study and the demanding work ahead to achieve effective artificial cognitive architecture in industrial environments, our subsequent objectives in the near future are to design a practical goal management procedure, to add further models to the repository for more complex tests, and to improve the way in which our instantiation can execute the components for improved performance on this new low-cost computing platform.

Acknowledgements

This work was supported by the Spanish Ministry of Economy and Competitiveness through its DPI2012-35504 CONMICRO project and FEDER funds. The authors wish also to thank Fernando Castaño for the support in the realization of experiments and processing data from experiments.

Annex I

Log file for the reinforcement leaning of the inverse model.

```
ANTICIPATION=8.332111986229894
5.0
Performance Index (MSE): 3.7769246696735976
Finishing drilling No.- 1
ORGANIZATION BEGINS
Performance Index (MSE): 6.020674866392445
Finishing drilling No.- 2
Performance Index (MSE):_learning: 5.696279190950438
NEXT ACTION: ExecutionNode(InverseAnfis(config/inverso_nuevo.ini)):
{Kin(DOUBLE)=0.748, meanOut(DOUBLE)=52.286} |
Performance Index (MSE): 5.696279190950438
Finishing drilling No.- 3
Performance Index (MSE):_learning: 3.3474918433763614
NEXT ACTION: ExecutionNode(InverseAnfis(config/inverso_nuevo.ini)):
{Kin(DOUBLE)=0.748, meanOut(DOUBLE)=59.286} |
ORGANIZATION BEGINS
Performance Index (MSE):Table: {ANTICIPATION=3.8459538576617147} Best:
ANTICIPATION
Performance Index (MSE): 3.3474918433763614
Finishing drilling No.- 4
Performance Index (MSE): 1.988387380851299
Finishing drilling No.- 5
ORGANIZATION BEGINS
Performance Index (MSE): 1.6155887571517007
Finishing drilling No.- 6
Performance Index (MSE):_learning: 1.2995309378912394
NEXT ACTION: ExecutionNode(InverseAnfis(config/inverso_nuevo.ini)):
{Kin(DOUBLE)=0.748, meanOut(DOUBLE)=66.286} |
Performance Index (MSE): 1.2995309378912394
Finishing drilling No.- 7
Performance Index (MSE):_learning: 1.6727367117380498
NEXT ACTION: ExecutionNode(InverseAnfis(config/inverso_nuevo.ini)):
{Kin(DOUBLE)=0.748, meanOut(DOUBLE)=73.286} |
ORGANIZATION BEGINS
Performance Index (MSE):Table: {ANTICIPATION=1.8900584263304163} Best:
ANTICIPATION
Performance Index (MSE): 1.6727367117380498
Finishing drilling No.- 8
Performance Index (MSE): 0.8872413435935801
Finishing drilling No.- 9
ORGANIZATION BEGINS
The last mode (ANTICIPATION was successful
Performance Index (MSE): 0.4017396251815698
Finishing drilling No.- 10
```

Log file for the organization.

```
{ANTICIPATION=0.8666028782473348, SINGLE LOOP=0.8771843826592491,
ANTICIPATION+MIRRORING=0.8653732453405826}
5.0
ORGANIZATION BEGINS
The last mode (SINGLE LOOP was successful
Performance Index (MSE): 0.45003959833218005
Finishing drilling No.- 1
Performance Index (MSE): 0.43637194274581015
Finishing drilling No.- 2
ORGANIZATION BEGINS
Performance Index (MSE)Table: {ANTICIPATION=0.8666028782473348, SINGLE
LOOP=1.0575288999477688, ANTICIPATION+MIRRORING=0.8653732453405826}
Best: ANTICIPATION+MIRRORING
Changing to ANTICIPATION+MIRRORING
Performance Index (MSE): 1.1202816580791999
Finishing drilling No.- 3
Performance Index (MSE): 0.9572464140536598
Finishing drilling No.- 4
ORGANIZATION BEGINS
Performance Index (MSE)Table: {ANTICIPATION=0.8666028782473348, SINGLE
LOOP=1.0575288999477688, ANTICIPATION+MIRRORING=0.7269170584936862}
Best: ANTICIPATION+MIRRORING
Performance Index (MSE): 0.7115330377329199
Finishing drilling No.- 5
Performance Index (MSE): 0.27312344216621987
Finishing drilling No.- 6
ORGANIZATION BEGINS
The last mode (ANTICIPATION+MIRRORING was successful
Performance Index (MSE): 0.2892327149771201
Finishing drilling No.- 7
Performance Index (MSE): 0.5820603007558
Finishing drilling No.- 8
ORGANIZATION BEGINS
The last mode (ANTICIPATION+MIRRORING was successful
Performance Index (MSE): 0.442942461819889 94
Finishing drilling No.- 9
Performance Index (MSE): 0.3704961754985
Finishing drilling No.- 10
ORGANIZATION BEGINS
The last mode (ANTICIPATION+MIRRORING was successful
Performance Index (MSE): 0.3704961754985
Finishing drilling No.- 11
```

References

[1] O.J. Romero Lopez, Self-organized and evolvable cognitive architecture for intelligent agents and multi-agent systems, in: C. DiChic, C. Cotta, M. Ebner, A. Ekart, A.I. Esparcia Alcazar, C.K. Goh, J.J. Merelo, F. Neri, M. Preuss, J. Togelius, G.N. Yannakakis (Eds.), Applications of Evolutionary Computation, Pt I, Proceedings, 2010, pp. 392–401.

[2] A. Bannat, T. Bautze, M. Beetz, J. Blume, K. Diepold, C. Ertelt, F. Geiger, T. Gmeiner, T. Gyger, A. Knoll, C. Lau, C. Lenz, M. Ostgathe, G. Reinhart, W. Roesel, T. Ruehr, A. Schuboe, K. Shea, I. Stork Genannt Wersborg, S. Stork, W. Tekouo, F. Wallhoff, M. Wiesbeck, M.F. Zaeh, Artificial cognition in production systems, IEEE Trans. Autom. Sci. Eng. 8 (2011) 148–174.

[3] A. Sanchez Boza, R. Haber Guerra, A. Gajate, Artificial cognitive control system based on the shared circuits model of sociocognitive capacities. A first approach, Eng. Appl. Artif. Intell. 24 (2011) 209–219.

[4] M. Bazhenov, R. Huerta, B.H. Smith, A computational framework for understanding decision making through integration of basic learning rules, J. Neurosci. 33 (2013) 5686–5697.

[5] M. Khamassi, S. Lallée, P. Enel, E. Procyk, P.F. Dominey, Robot cognitive control with a neurophysiologically inspired reinforcement learning model, Front. Neurorobotics (2011) 1–14.

[6] D. Bruckner, H. Zeilinger, D. Dietrich, Cognitive automation survey of novel artificial general intelligence methods for the automation of human technical environments, IEEE Trans. Ind. Inform. 8 (2012) 206–215.

[7] S. Borgo, An ontological approach for reliable data integration in the industrial domain, Comput. Ind. 65 (2014) 1242–1252.

[8] M. Fatemi, S. Haykin, Cognitive control: theory and application, IEEE Access 2 (2014) 698–710.

[9] J. Albus, Toward a computational theory of mind, J. Mind Theory. Rigor Cogn. Sci. (2009) 1–38.

[10] T. Meystel, On intelligence control, learning and hierarchies, IEEE Control Syst. Mag. (1994) 63–74.

[11] J.R. Anderson, D. Bothell, M.D. Byrne, S. Douglass, C. Lebiere, Y. Qin, An integrated theory of the mind, Psychol. Rev. (2004) 1036–1060.

[12] A. Newell, Unified Theories of Cognition, Harvard University Press, 1994.

[13] P. Langley, J.E. Laird, S. Rogers, Cognitive architectures: research issues and challenges, Cogn. Syst. Res. 10 (2009) 141–160.

[14] M. Rabinovich, I. Tristan, P. Varona, Neural dynamics of attentional cross-modality control, PLOS ONE 8 (2013).

[15] R.R. Llinas, S. Roy, The 'prediction imperative' as the basis for self-awareness, Philos. Trans. R. Soc. B Biol. Sci. (2009) 1301–1307.

[16] M. Ito, Control of mental activities by internal models in the cerebellum, Nat. Rev. Neurosci. (2008) 304–313.

[17] S. Hurley, The shared circuits model (SCM): how control, mirroring, and simulation can enable imitation, deliberation, and mindreading, Behav. Brain Sci. (2008), 1–22, 52–58.

[18] M. Minsky, The Emotion Machine, 2006.

[19] S. Kopácsi, G.L. Kovács, J. Nacsa, Some aspects of dynamic 3D representation and control of industrial processes via the Internet, Comput. Ind. 64 (2013) 1282–1289.

[20] H. Imamizu, M. Kawato, Brain mechanisms for predictive control by switching internal models: implications for higher-order cognitive functions, Psychol. Res. 73 (2009) 527–544.

[21] A. Sanchez Boza, R. Haber Guerra, A First Approach to an Artificial Networked Cognitive Control System Based on the Shared Circuits Model of Sociocognitive Capacities, Connectionist Models of Behavior and Cognition, World Scientific Publishing, 2011.

[22] R.-E. Precup, P. Angelov, B.S.J. Costa, M. Sayed-Mouchaweh, An overview on fault diagnosis and nature-inspired optimal control of industrial process applications, Comput. Ind. (2015).

[23] N. Chungoora, R.I. Young, G. Gunendran, C. Palmer, Z. Usman, N.A. Anjum, A.-F. Cutting-Decelle, J.A. Harding, K. Case, A model-driven ontology approach for manufacturing system interoperability and knowledge sharing, Comput. Ind. 64 (2013) 392–401.

[24] J.I.M. Carpendale, C. Lewis, Mirroring cannot account for understanding action, Behav. Brain Sci. (2008) 23–24.
[25] T. Makino, Failure, instead of inhibition, should be monitored for the distinction of self/other and actual/possible actions, Behav. Brain Sci. (2008) 32–33.
[26] D. Kit, D.H. Ballard, B. Sullivan, C.A. Rothkopf, A hierarchical modular architecture for embodied cognition, Multisens. Res. 26 (2013) 177–204.
[27] M.E. Bratman, D.J. Israel, M.E. Pollack, Plans and Resource-Bounded Practical Reasoning, John Wiley & Sons, 1988, pp. 349–355.
[28] P. Langley, Cognitive architectures and general intelligent systems, AI Mag. (2006) 33–34.
[29] J.F. Lehman, J. Laird, P. Rosenbloom, A Gentle Introduction to Soar, An Architecture for Human Cognition, 1998, 211–253.
[30] R. Sun, E. Merrill, T. Peterson, From implicit skills to explicit knowledge: a bottom-up model of skill learning, Cogn. Sci. (2001) 203–244.
[31] R. Sun, X. Zhang, Accounting for a variety of reasoning data within a cognitive architecture, J. Exp. Theor. Artif. Intell. (2006) 169–191.
[32] Z. Mathews, S. Bermudez i Badia, P.F.M.J. Verschure, PASAR: an integrated model of prediction, anticipation, sensation, attention and response for artificial sensorimotor systems, Inf. Sci. 186 (2012) 1–19.
[33] S. Franklin, T. Madl, S. D'Mello, J. Snaider, LIDA: a systems-level architecture for cognition, emotion, and learning, IEEE Trans. Auton. Ment. Dev. 6 (2014) 19–41.
[34] V. Cutsuridis, J.G. Taylor, A cognitive control architecture for the perception-action cycle in robots and agents, Cogn. Comput. 5 (2013) 383–395.
[35] D. Vernon, G. Metta, G. Sandini, A survey of artificial cognitive systems: implications for the autonomous development of mental capabilities in computational agents, IEEE Trans. Evol. Comput. (2007) 151–180.
[36] A.S. Boza, R.H. Guerra, A first approach to artificial cognitive control system implementation based on the shared circuits model of sociocognitive capacities, ICIC Express Letters 4 (2010) 1741–1746.
[37] L.P. Kaelbling, M.L. Littman, A.W. Moore, Reinforcement learning: A survey, Journal of Artificial Intelligence Research 4 (1996) 237–285.
[38] C. Szepesvári, Algorithms for reinforcement learning, Synthesis Lectures on Artificial Intelligence and Machine Learning 4 (2010) 1–103.
[39] Y. Wang, Y. Yang, Particle swarm optimization with preference order ranking for multi-objective optimization, Information Sciences 179 (2009) 1944–1959.
[40] J. Zhang, J. Zhuang, H. Du, S. Wang, Self-organizing genetic algorithm based tuning of PID controllers, Information Sciences 179 (2009) 1007–1018.
[41] I. Mukherjee, P.K. Ray, A review of optimization techniques in metal cutting processes, Comput. Ind. Eng. 50 (2006) 15–34.
[42] R.E. Haber, R.M. Del Toro, A. Gajate, Optimal fuzzy control system using the cross-entropy method. A case study of a drilling process, Inf. Sci. 180 (2010) 2777–2792.
[43] E. Zitzler, K. Deb, L. Thiele, Comparison of multiobjective evolutionary algorithms: empirical results, Evol. Comput. 8 (2000) 173–195.
[44] S. Huband, P. Hingston, L. Barone, L. While, A review of multiobjective test problems and a scalable test problem toolkit, IEEE Trans. Evol. Comput. 10 (2006) 477–506.
[45] R.E. Precup, R.C. David, E.M. Petriu, M.B. Radac, S. Preitl, Adaptive GSA-based optimal tuning of PI controlled servo systems with reduced process parametric sensitivity, robust stability and controller robustness, IEEE Trans. Cybern. 44 (2014) 1997–2009.
[46] R.E. Precup, R.C. David, E.M. Petriu, S. Preitl, M.B. R?dac, Fuzzy logic-based adaptive gravitational search algorithm for optimal tuning of fuzzy-controlled servo systems, IET Control Theory Appl. 7 (2013) 99–107.
[47] S. Barchinezhad, E. Mahdi, A new fuzzy and correlation based feature selection method for multiclass problems, Int. J. Artif. Intell. 12 (2014) 24–41.
[48] T. Chen, K. Tang, G. Chen, X. Yao, Analysis of computational time of simple estimation of distribution algorithms, IEEE Trans. Evol. Comput. 14 (2010) 1–22.
[49] R. Rubinstein, A stochastic minimum cross-entropy method for combinatorial optimization and rare-event estimation, Methodol. Comput. Appl. Probab. 7 (2005) 5–50.
[50] R. Rubinstein, Semi-iterative minimum cross-entropy algorithms for rare-events, counting, combinatorial and integer programming, Methodol. Comput. Appl. Probab. 10 (2008) 121–178.
[51] H. Boubertakh, M. Tadjine, P.-Y. Glorennec, S. Labiod, Tuning fuzzy PD and PI controllers using reinforcement learning, ISA Trans. 49 (2010) 543–551.
[52] S. Haykin, M. Fatemi, P. Setoodeh, Y. Xue, Cognitive control, Proc. IEEE 100 (2012) 3156–3169.
[53] K. Park, L. Heuiseok, Design and implementation of knowledge sharing system using smart devices, Int. J. Artif. Intell. 13 (2015) 1–7.
[54] D.C. Schmidt, D.L. Levine, S. Mungee, The design of the TAO real-time object request broker, Comput. Commun. 21 (1998) 294–324.
[55] M. Henning, Choosing Middleware: Why Performance and Scalability Do (and Do Not) Matter, 2009.
[56] J.F. Koning, C.J.M. Heemskerk, P. Schoen, D. Smedinga, A.H. Boode, D.T. Hamilton, Evaluating ITER remote handling middleware concepts, Fusion Eng. Des. 88 (2013) 2146–2150.
[57] Raspberry Pi model B specifications, http://docs-europe.electrocomponents.com/webdocs/127d/0900766b8127da4b.pdf.
[58] CNET: Create a retro game console with the Raspberry Pi, http://www.cnet.com/how-to/create-a-retro-game-console-with-the-raspberry-pi.
[59] Meet Jasper: open-source voice computing, http://www.raspberrypi.org/meet-jasper-open-source-voice-computing.
[60] Unified Modeling Language, http://www.uml.org.
[61] Real Time Specification for Java, http://www.rtsj.org.
[62] S.W.a.I. Generator, http://www.swig.org/index.php.
[63] G. Beruvides, R. Quiza, R. del Toro, F. Castaño, R. Haber, Correlation of the holes quality with the force signals in a microdrilling process of a sintered tungsten–copper alloy, Int. J. Precis. Eng. Manuf. 15 (2014) 1801–1808.
[64] J.-S.R. Jang, ANFIS: adaptive-network-based fuzzy inference system, IEEE Trans. Syst. Man Cybern. 23 (1993) 665–685.
[65] M. Pratama, S.G. Anavatti, P.P. Angelov, E. Lughofer, PANFIS: a novel incremental learning machine, IEEE Trans. Neural Netw. Learn. Syst. 25 (2014) 55–68.
[66] R.E. Precup, H. Hellendoorn, A survey on industrial applications of fuzzy control, Comput. Ind. 62 (2011) 213–226.

Rodolfo E. Haber received the Ph.D. degree in Industrial Engineering from the Universidad Politécnica de Madrid, Madrid, Spain, in 1999. He is with the Centre of Automation and Robotics-CAR, CSIC-UPM (formerly Institute for Industrial Automation – IAI) since 1996 and has actively participated in a number of National, European and International RTD projects in the area of control systems. In 1999, he also joined the Computer Engineering Department at the Universidad Autónoma de Madrid as an adjunct professor. He has over 100 technical papers, in the areas of the application of intelligent systems to complex processes. He has taught several seminars on the same theme in the Georgia Institute of Technology and University of Central Florida. His research interests include networked and embedded intelligent control systems, artificial cognitive control systems, modeling, control and supervision of complex processes.Dr. Haber was an IPC member for several IEEE and IFAC conferences since 2000. Since 2002 he has belonged to IFAC's Technical Committee 3.1 *Computers for Control*. Since 2005 he has been member of the ASME Technical Committee for *Model Identification and Intelligent Systems*. Since 2003 he has served as expert evaluator for the European Commission in the *6th and 7th Framework Programme*. Since 2007, he has been associate editor of the *Transactions on Computational Science* and the *International Journal of Mechatronics and Manufacturing Systems*.

Carmelo Juanes was born in 1991 in Linares, Spain. He obtained the double degree of Bachelor of Computer Science and Mathematics at Autonomous University of Madrid (UAM) in 2014. In 2010 he joined the research group C4LIFE at Escuela Politécnica Superior. Since September, 2014 he is a PhD candidate at the Center for Automation and Robotics (UPM-CSIC) in the field of artificial cognitive systems. His main interests are intelligent systems, self-adaptive systems and self-organized, advanced strategies of software engineering.

Raúl M. del Toro received his B.E. degree in Automation Engineering from the Universidad de Oriente, Santiago de Cuba, in 1997, the M.S. degree in automation and robotics from the Polytechnic University of Madrid in 2009 and the Ph.D degree in Computer Science & Telecommunication Engineering from the Universidad Autónoma de Madrid in 2011. In 2006 he joined to the Spanish Council for Scientific Research (CSIC), where he is currently a researcher at the Centre of Automation and Robotics-CAR (formerly Institute for Industrial Automation) working on several research and development projects. His research interests include intelligent control systems, modeling, control and monitoring of complex electromechanical processes.

Gerardo Beruvides graduated with a B.Sc. (cum laude), in 2010, and a M.D. in Mechanical Engineering, in 2012, from the University of Matanzas (Cuba), where he is currently Assistant Professor. He was a recipient of Ph.D. grant supported by Spanish Ministry of Economy and Competitiveness through its DPI2012-35504 CONMICRO project. His research interests focus on the modeling and optimization of machining process and applied artificial intelligence. He has published several papers on these topics and he is reviewer of several international journals.