SAMBA: A Self-Aware Health Monitoring Architecture for Distributed Industrial Systems

Lydia C. Siafara*, Hedyeh A. Kholerdi*, Aleksey Bratukhin†, Nima TaheriNejad*, Alexander Wendt*,
Axel Jantsch*, Albert Treytl† and Thilo Sauter*†
* Institute of Computer Technology, TU Wien, Vienna, Austria
† Center for Integrated Sensor Systems, Danube University Krems, Wr. Neustadt, Austria
Email: *{first name.surname}@tuwien.ac.at, †{first name.surname}@donau-uni.ac.at

***Abstract*—In the context of Industry 4.0, constantly evolving shop floors generate the need for a highly adaptive and autonomous automation system with lean maintenance, minimum downtime, maximum reliability, and resilience. Future Manufacturing Execution Systems (MESs) will be more complex and dynamic as well as distributed physically and logically. This makes it very difficult, if not impossible, for the conventional centralized architectures to effectively control these vibrant Cyber-Physical Production Systems (CPPSs). To address these issues, we propose Self-Aware health Monitoring and Bio-inspired coordination for distributed Automation systems (SAMBA), an architecture which tackles these challenges. SAMBA increases the ability of the system to intelligently adapt to rapidly changing environment and conditions of future CPPSs.**

***Index Terms*—Self-awareness, Monitoring, Health Monitoring, Distributed Cyber-Physical Systems, Bio-inspired, Cognitive System, Automation Systems, System Architecture.**

I. Introduction

Modern industrial systems are complex systems that integrate distributed physical, software and, network components, i.e., Cyber-Physical Production Systems (CPPSs). Continuous advances in all three areas of hardware, software, and communication increase the expectations on the performance of these systems in terms of flexibility, reliability, and responsiveness. To address the requirements, there is a shift towards distributed design solutions that give more autonomy to field level controllers. This interest is further reflected in the decentralized decision design principles of Industry 4.0 [1]. To maintain the operability of such heterogeneous and dynamically evolving systems, there is a need for intelligent devices with self-monitoring and self-maintenance properties [2]. Autonomous self-organization systems, common in real life organisms, are hence a focal point in Bio-Inspired Artificial Intelligence [3].

A major challenge in distributed industrial systems is to cope with the distributed nature of events and the lack of central interventions, in a dynamic environment with constantly increasing volume of interconnected devices [2]. The growing complexity makes the maintenance of a model that captures the system dynamics unfeasible [4]. Therefore, the application of model-based monitoring techniques is also not a suitable solution [5]. In addition to that, real-time constraints generate the need for fast, coordinated diagnostics and decision-making for quick detection and handling of events and anomalies during the operation. Cognitive systems, inspired by the method of processing information by human, can process information efficiently in higher levels of abstractions while they learn to adapt their behavior to the dynamics of the environment [6].

In this work, we propose an architecture for supervision of distributed, dynamic, and complex CPPSs. The proposed system is a community of individual Autonomous Cooperating Objects (ACOs). Each ACO in the system learns the specifics of its environment locally and makes decisions to pursue its own goals regarding behavior and performance. Further, each entity is only aware of its individual goals whereas altogether the system pursues the global objectives which are not represented explicitly in any individual entity. The architecture considers self-aware local data analysis and cognitive decision-making to detect changes, anomalies, and unexpected events during the operation, to adapt accordingly. Through the mitigation of errors, it reduces costs imposed by downtime and maintenance interventions.

In the rest of this paper, we first review the state of the art in the areas relevant to our approach, i.e., autonomous monitoring, cognitive systems and distributed clustering. In Section III, we explain how these methods are integrated into the Self-Aware health Monitoring and Bio-inspired coordination for distributed Automation systems (SAMBA) architecture and present the modules of the solution and their main functionality. Next, we go through two exemplary scenarios with anomalies and elaborate the system behavior for handling these events in Section IV. Finally, we discuss potential benefits of the proposed architecture and draw our conclusions.

II. State of the art

There is a strong need for today's automation systems to become smart, where smart is defined by extended functionality, multi-functionality, self-diagnosis, and reconfigurability [7]. Smart applications in this context are usually systems with distributed intelligence. However, the controllers normally use a legacy or proprietary hardware and software environment [8], which increases the complexity of reconfiguration. In subsystems within the smart applications, several processes are performed in parallel, which further increase the overall complexity of the system. This also makes software engineering more difficult because there is a strong dependency on vendors [9].

Among recently developed platforms Guang et al. [10], [11] have pursued self-monitoring and adaptation as first-class citizens in their systems. To that end, they have devised a general, agent-based architecture and design methodology, namely HAMSoC. Sarma et al. [12], [13] take the view that self-awareness is dependent on monitoring, measuring various kinds of data and working on virtual sensing through processing existing hardware measurement data. Upon their platform, a sophisticated, self-aware system is based. Hoffmann and co-workers have developed SEEC [14], a general framework for self-awareness using an observe-decide-act paradigm. The system cyclically monitors key features, applies a control and decision algorithm, and deploys appropriate actions to adapt to changes in the environment and its own state. Awareness may also emerge in cooperating systems of systems (SoS). Preden and coworkers [15], [16] have studied distributed surveillance systems and assign particular importance to the role of attention and context awareness in sensing and processing data. They believe that these properties facilitate the efficient operation of distributed sensing systems.

A challenge of software design is to design a decision-making module that is able to respond to different types of inputs with different sequences of actions. "Cognition can be viewed as the process through which the system achieves robust, adaptive, anticipatory, autonomous behavior, entailing perception and action" [17]. In a technical application, the functionality of a cognitive system can be broken down to a map between situations and corresponding actions that satisfy goals [18]. Because they are inspired by biological systems, their natural application is in robots like SOAR [19] and BDI [20]. Some of other applications are virtual actors in simulators like ICARUS [21], TAC-Air-Soar [19] and BDI [22]. The US Navy also uses cognitive systems, e.g., the LIDA architecture [23] to manage jobs for sailors. The system offers jobs depending on the sailor's preferences, the Navy's policies, the needs of the tasks and their priorities. The cognitive model SiMA [24] mimics the human psyche by providing functions that generate human behavior. Unfortunately, among the many of the known cognitive architectures, most of them have never left the laboratory. An attempt to do this is made in the project KORE [25], which applies the SiMA model to a generic human-inspired control system that learns from experience and generates rule-based control strategies for building automation. The application of KORE serves as a starting point for the decision-making module in SAMBA.

One of the major challenges in distributed industrial systems is the provision of a global overview that reflects the overall system objectives without compromising the autonomous behavior of system components [26], [27], [28]. A relevant approach to distributed industrial automation is the PROSA architecture [29]. It addresses the global overview problem via dynamic clustering of tasks that can be resolved more efficiently from a central control point. For a particular problem, it assigns a temporary mediator with a limited scope of the control functionality, [30], [31]. While dynamic clustering algorithms are still very uncommon in industrial systems, they have been used extensively in ad-hoc wireless networks [32] mostly for optimizing the routing protocol [33] with energy consumption and delay as main objective functions. Interestingly, in search for adaptive, scalable, and robust solutions a substantial body of work has linked clustering and routing in wireless sensor networks with concepts based on swarm intelligence [34]. However, what all of these solutions have in common is introducing a central point of control. The proposed solution aims to avoid this by introducing local goal propagation instead of assigning a hierarchy.

III. Proposed Architecture

This section describes a high-level overview of the proposed architecture for self-aware health monitoring in distributed CPPS and some of its functionalities.

A. Definitions

In the rest of this paper following terms are used which we define them here: *component* refers to sensors, actuators, and other similar soft/hardware parts in a device. *Entity* refers to a set of components (hardware and software). *Autonomous Cooperating Object (ACO)* refers to the entity which has (and can independently complete) a task concretely defined in the production process [35]. An ACO processes the data from its components, controls its components, performs problem mitigation regarding its task, and deals with other ACOs through negotiation. There is often a one to one correlation between entities and the ACO logic added by SAMBA. *Human* refers to a human being (e.g., operator) in the industrial systems. *System* refers to a set of entities and, if applicable, humans. *Product Object* refers to a manipulated object, not belonging to any entity in the system, which consists of parts or the entirety of the final product of the system.

B. System Architecture

An entity of the system has a specific functionality in the production process; it includes hardware parts, a control unit and a logical unit which is often encapsulated as an ACO. The ACO interacts with the environment through the hardware parts, i.e., its sensors and actuators, and provides the set-points to the controller for regulating the behavior of the entity locally. As a result, the global system behavior emerges from the local decisions of these distributed entities.

The general architecture of an entity is illustrated in Fig. 1. The main components of the ACO are the operation module, Self-Aware Health Monitoring (SAHM), internal manager and, external manager. The role of the operation module is to transfer the sensed data to the SAHM and the internal manager. The SAHM detects and diagnoses faults and failures in the system. When such events occur, the internal manager uses appropriate reasoning to find and execute the best policy for the detected errors. If this policy involves other ACOs in the cluster, the external manager negotiates with them in this regard; otherwise, it informs them about relevant changes.

During the system operation, the operation module consistently updates the ACO with information from its local

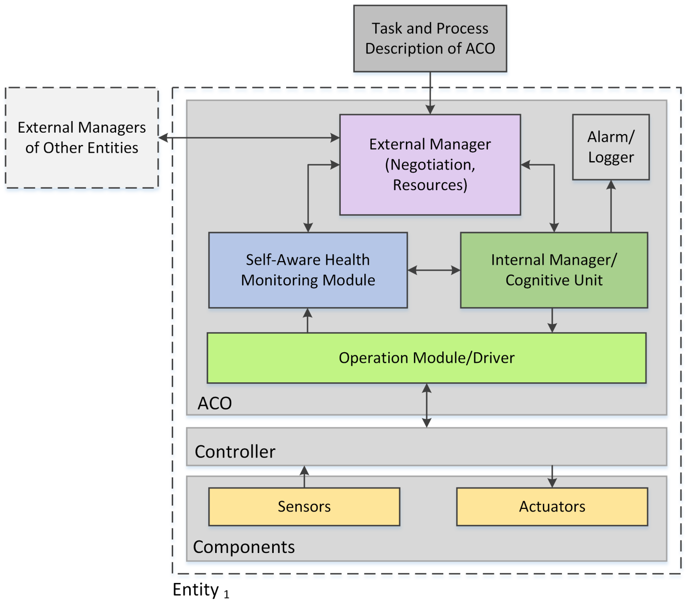
Task and Process Description of ACO
External Managers of Other Entities
External Manager (Negotiation, Resources)
Alarm/ Logger
Self-Aware Health Monitoring Module
Internal Manager/ Cognitive Unit
Operation Module/Driver
ACO
Controller
Sensors
Actuators
Components
Entity 1

Fig. 1. General Architecture of an entity.

environment. It shares the data not only with the SAHM and the internal manager of its entity but also with other requesting ACOs (indirectly and through external manager). The SAHM monitors the health status of the ACO by processing information from the operation module, and if necessary, by requesting information from other ACOs to interpret the local information. This module is working independently from the internal manager, which is the decision-making unit of the ACO. The internal manager receives the requested abstracted data and the health status from the SAHM, and a list of available resources and the requests of other ACOs through the external manager; it evaluates the goals and chooses the actions that maximize the chances of fulfilling its goals. Also, if an error is detected by the SAHM, it reports it to the logger and, if human intervention is necessary, it notifies the operator of the system (user). Finally, the external manager is responsible for responding to requests from the internal manager and the SAHM, and for establishing connections with the other ACOs of the cluster. It also keeps a list of external resources (other ACOs) in the cluster which can help the operation. Therefore, it is aware of the existence of its neighbors, their types and their order in the process. The purpose of each subsystem in the architecture is discussed in more detail in rest of this section.

1) Operation Module: which can be interpreted to be a driver, translates the data between the hardware and software parts of a system. The operation module serves as an interface between hardware and software to communicate with the physical environment and provide information for SAHM. It also serves (when necessary) as the intermediating agent between the legacy systems and the ACO. Moreover, it receives the high-level set-points from the internal manager and translates the to the low-level settings of its components.

2) SAHM: aims at data abstraction and anomaly detection inspired by natural phenomena and self-awareness concepts such as history, data reliability, attention, and confidence. For example, using history in a self-reflective manner, it creates better abstractions and finds methods which have a better performance and a higher confidence. Anomaly detection is split into four sub-modules which detect a specific information of the anomaly including anomaly itself, its characteristics, and its cause as well as potential future effects of the anomaly. Here, SAHM uses attention to trigger certain processes (e.g., prediction, or cause detection) only when necessary, and confidence to trigger only the best algorithms (proven to be so for the current condition according to the history). Thus, it can maintain a small footprint without limiting its performance. The detection approach mostly relies on the self-observation, however, sometimes negotiation with other entities increases the knowledge of an entity regarding itself and anomalies. Therefore, the self-aware health monitoring module monitors various parameters of a subsystem in a distributed manner.

3) Internal Manager: The decision-making problem in an ACO is formulated as the problem to select one action that satisfies system objectives and fulfills the constraints of the external environment. Here, system objectives are defined by three inputs: system performance, health value, and requests from other ACOs. The system performance is estimated on the basis of the production plan of the Manufacturing Execution System (MES), whereas the health value is provided by the SAHM as a means of how error-free the system is currently working. Requests from other ACOs are provided by the external manager. The internal manager decides, which of the system objectives shall be given the priority and what the ACO shall do to satisfy them, e.g., mitigate an error in the entity if the health value is low. Methods used are case-based reasoning and reinforced learning, based on the categorized causes of problems discovered by SAHM. Through information requests from other ACOs, the lack of an accurate model in a complex system is compensated. The internal manager improves its behavior through the evaluation of its previously executed actions while taking into account the system dynamics. Further, it also needs to ensure that safety requirements and timing constraints are always met.

4) External Manager: In order to compensate for the lack of a global overview due to the distributed nature of the proposed system, dynamic clustering is applied, which provides a flexible way to integrate global objectives in a distributed environment. The challenge is to form clusters based on functional requirements of the shop floor and manufacturing instructions. Currently, there is no methodology to formalize production order ontology in relation to the anomalies. Instead of pre-defined rules of the existing solutions, the external manager defines dynamic methods using learning algorithms (e.g., neural networks, and reinforcement learning) to automatically discover the cluster formation rules in the basis of the production order and the physical layout of the shop floor. A set of generic classifiers that operate on non-application-specific characteristics are defined and used by the external manager to determine the essential connections in the cluster and further establish weighted links between ACOs based on the prioritized set of controlled parameters. These weighted connections are updated during run-time using backpropagation based on the results of communication and decision execution evaluations, providing an additional level of flexibility and adaptation to the system. The resulting unit, external manager, provides an implicit connectivity among the relevant ACOs and facilitates transparent negotiations between SAHM and the internal manager of one ACO and those of other ACOs in the cluster via establishing their relevance to a particular request. The external manager bases its technique on a non-deterministic relevance of connection between individual ACOs about a particular event.

IV. Use Case Example

To understand how the proposed architecture works, we go through a use case example. The error mitigation behavior is applied in the domain of conveyor systems.

A. Specification

The use case describes the operation of four conveyor sections that supply two robotic arms with objects. A potential configuration of the system is illustrated in Figure 2.

There are three different types of entities in the system: i) *Conveyor sections:* Each section includes the conveyor belt and supporting equipment, such as the motor connected to it, along with the sensors for object detection and monitoring of the system parameters (e.g., current, voltage of the motor, and velocity), ii) *Product sorter:* It includes the sorting mechanism and the sensors for sensing the objects for sorting. Depending on the object it supplies the conveyor belts 2, 3 and 4 accordingly, iii) *Robotic arms:* Each arm has (a) motor(s) and sensors to detect the position of the item. Depending on the task completion time they generate a supply rate demand and send it to the neighboring conveyor belts.

The central conveyor (conveyor 1) transports objects that according to selection criteria are either routed by the product sorter to conveyors 3 or 4 for processing, or to conveyor 2 (for human intervention or further processing). Each conveyor section represents an entity; it is controlled separately by the logical unit (ACO) and is connected to sensors and actuators (e.g., a motor, or other actuators). Similarly, the sorter and the robotic arms are separate entities. Since there is no central control unit, the different entities communicate with the neighboring entities and adjust their behavior accordingly to make the whole system operate effectively.

The robotic arms 1 and 2 are processing two different products. Given the processing time needed for the tasks each arm communicates with the neighboring conveyor section and creates a request for the expected supply rate. These requests are sent to the neighboring conveyors 3 and 4, which in their turn send each one separately a request for throughput to the conveyor 1. Conveyors 1-4, are then adapting their speeds accordingly to satisfy the required throughput rates.

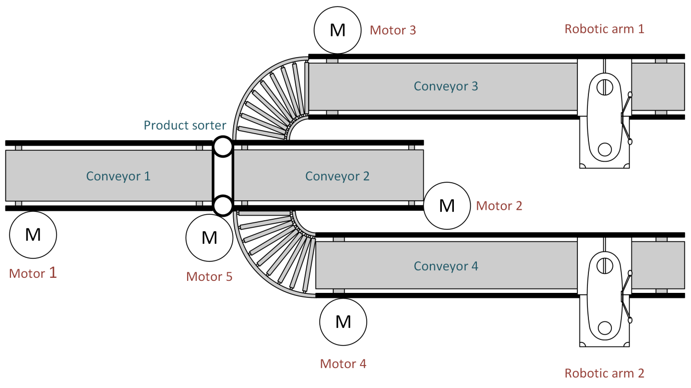

Fig. 2. SAMBA use case with conveyor belts and robotic arms.

B. Potential Problems

Different problems may occur during the operation of the system due to faulty equipment, false measurements, mis-tuning, failed quality controls, missing products, and communication errors. Among others, we present following problems which we will elaborate in this paper and later in this section, we will describe how SAMBA can address them. We use as a starting point two potential scenarios beyond the nominal operation, which the system needs to address properly: i) a motor wearing out, and ii) a change of speed request. We note that this is not an exhaustive nor an exclusive list of all the possible scenarios. The purpose of this section, however, is to explain how the resulting system is able to handle unexpected events that might appear during the system operation.

In the wearing out of a motor scenario the ACO identifies the problem in the involved entity, it evaluates the priority of the problem and adapts the settings at this entity. If these adaptations lead to changes in the external behavior of the entity, then the system informs other relevant entities as well to synchronize the system. In the change of operating speed scenario, an entity decides to change the speed and modifies the number of objects it can process over time. When variation in the operating speed happens, the system should inform other entities to update themselves, so that the whole system behavior remains synchronized. In the rest of this chapter, we explain the system behavior that is triggered in each scenario.

C. SAMBA Solutions

To explain the SAMBA solution, we specify here the functions that generate the required system behavior in the presence of the aforementioned problems. This analysis shows how each of the tasks falls into the responsibility definition of each module and how it is handled. To this end, we examine each scenario separately and specify the high-level behavior of the ACO. In wearing out of a component:

1) SAHM identifies the problem with a degree of certainty, estimates the deviation from normal performance, makes a prediction about the expected lifetime of the component and notifies the internal manager.

2) The internal manager evaluates the priority of the problem and adapts the policy of the ACO accordingly (e.g., by changing the set-point to reach the desired speed despite the wear-out).
3) The external manager upon request from the internal manager passes the necessary information to other ACOs of the cluster (e.g., in case they need to adapt to this change -i.e., uncompensated wear-out which is a slow down-, or if the ACO is leaving the cluster). If necessary, negotiation will take place to ensure the optimal operation of the larger system through the communication between the external managers of the involved ACOs.

Upon request for the change of speed, the following occurs:

1) The internal manager of the ACO initiating the change, we call it here ACO$_i$, notifies the external manager about the speed change event.
2) The external manager of ACO$_i$ contacts other related ACOs to make sure they are able to operate at this speed.
3) The external managers of the involved ACOs communicate with each other and send feedback to the requesting ACO$_i$. Negotiation is necessary to coordinate the collaboration of the ACOs and therefore, to minimize errors in the global system operation.
4) The external manager of the requesting unit, ACO$_i$, notifies its internal manager about the received feedback.
5) The internal manager of the ACO$_i$ decides to change (or not to change) the speed and sends a possible operational range to the external manager.
6) The external manager of ACO$_i$ sends the proposed operational range to the external manager of other ACOs.
7) The internal managers of other relevant ACOs are notified and decide whether to accept or reject the proposal. If they accept it, they specify the desired speed changes that are compatible with the proposal.
8) If a speed change is not an optional decision and shall occur in any case, the ACO$_i$ changes its speed, and the external managers of the dependent ACOs are notified (and they, in turn, notify their internal managers or those ACOs observe the event themselves).
9) The internal manager of ACO$_i$ adapts the speed based on the taken decision and sends the necessary command to the operation module (i.e., the controller).

If for some reason negotiation shall not (e.g., if the internal manager has to take such a decision independently due to certain constraints or problems) or cannot take place (communication between ACOs is interrupted), then only the last two steps are executed in this scenario.

V. Discussion

SAMBA is designed to operate on the top of existing systems without requiring major hardware changes, therefore, it can be applied to legacy systems with low investment costs. By adding self-awareness, it creates a self-adaptive and self-optimizing fault diagnosis and prognosis system able to both reliably improve itself while keeping a small footprint. A feature that is -to a large extent- missing in the current practice, and especially in the planned manner. In the proposed architecture the complexity of the supervising task is split into smaller problems that are solved locally. The local monitoring makes the error diagnosis and root cause analysis easier. Further, the lack of central supervision increases the flexibility and scalability of the system. This lack of central supervision is compensated through the ability to discover the environment and establish new clusters and collaboration with other ACOs, which dramatically reduces the design and engineering efforts compared to the state of the art.

The resulting system is expected to be able to detect, predict and mitigate potential health problems such that the downtime and number of necessary interventions are minimized. Additionally, the automatic reaction supports better scheduling of predictive maintenance by taking intermediate actions. Compared to the state of the art, the architecture introduces a new design paradigm in the industrial environment and CPPS that allows integration of legacy systems in manufacturing with considerably less effort due to the generic nature of the proposed architecture. It also provides enhanced scalability, especially in the ability of the systems to discover their environment and anomalies automatically, which dramatically reduces engineering efforts currently required. Furthermore, continuous feedback and learning in all three core functionalities (anomaly detection, environment discovery and decision making) improve the performance and flexibility of the system in dynamic manufacturing environments.

Another distinguishing novel characteristic of the proposed health monitoring unit (SAHM) is that it takes advantage of self- and context-awareness concepts to locally monitor the ACO's own status (enhanced by self-awareness concepts such as confidence, attention, data-reliability, history, relevance, and desirability [36], as well as the local context [37]). It complements this information through communication with other ACOs to obtain awareness about the bigger picture and the context in which observations are happening. The novelty of the decision-making unit (internal manager) is the adaptation of an existing cognitive architecture to a technical application that differs from simulated actors or other human-like applications. The use of cognitive methods provides an extra level of flexibility in finding solutions in the face of conflicting constraints. The communication unit (external manager) provides an implicit connectivity among the relevant ACOs via distributed clustering that takes into account the production order structure as well as the physical layout of the shop floor. The unit facilitates negotiations between the health monitoring and decision-making units of one ACO and those of other ACOs in the cluster via establishing their relevance to a particular request. Finally, the operation unit (operation module) which is the actual plant controller, or sits on top of the plant controller, enables the system to work with legacy systems smoothly. Thus, the proposed architecture does not necessarily need to modify the existing (legacy) systems but rather extends their operation by providing the connectivity and interoperability between the existing system and the aforementioned three other units to accomplish the set objectives. We note that the operation unit is still responsible for meeting safety requirements, and that other units are not allowed to initiate actions violating safety.

Next steps in the development of SAMBA include the detailed design of the algorithms for the modules of the architecture and testing of the emergent system performance. The algorithms need to run on embedded devices of industrial scale with soft real-time capabilities. Thus, they should be designed to address the small footprint and real-time constraints.

VI. Conclusion

In this paper, we proposed an architecture for industrial distributed CPPS to increase the ability of the system to intelligently adapt to changing environment and conditions. The main focus of this architecture is health monitoring and the mitigation of behavioral deviations. The system automatically detects the environment, forms clusters, analyzes and detects events, makes predictive decisions based on multiple non-predefined data processes and mitigates anomalies. The main benefits of the proposed architecture are showcased in four main system components and in the synergy they provide.

Acknowledgment

We gratefully acknowledge the financial support provided to us by the BMVIT/FFG under the program *ICT of the Future* in the project SAMBA (contract number: 855426).

References

[1] M. Hermann, T. Pentek, and B. Otto. Design principles for industrie 4.0 scenarios. In *System Sciences (HICSS), 2016 49th Hawaii Int. Conf.*, pages 3928–3937. IEEE, 2016.

[2] European Factories of the Future Research Ass. et al. Factories of the future: Multi-annual roadmap for the contractual ppp under horizon 2020. *Pub. office of the European Union: Brussels*, 2013.

[3] D. Floreano and C. Mattiussi. *Bio-inspired artificial intelligence: theories, methods, and technologies*. 2008.

[4] L. H. Chiang, E. L. Russell, and R. D. Braatz. *Fault detection and diagnosis in industrial systems*. Springer, 2000.

[5] S. Yin, X. Li, H. Gao, and O. Kaynak. Data-based techniques focused on modern industry: An overview. *IEEE Transactions on Industrial Electronics*, 62(1):657–667, 2015.

[6] Y. Wang. On cognitive learning methodologies for cognitive robotics. In *15th Int. Conf. on Robotics, Control and Manufacturing Technology*, pages 11–19, 2015.

[7] T. Bangemann, S. Karnouskos, R. Camp, O. Carlsson, M. Riedl, S.t McLeod, R. Harrison, A. W. Colombo, and P. Stluka. State of the art in industrial automation. In *Industrial Cloud-Based CPS*, pages 23–47. Springer, 2014.

[8] R. M. da Silva, F. Junqueira, D. J. dos Santos Filho, and P. E. Miyagi. Design of reconfigurable and collaborative control system for productive systems. 2012.

[9] B. Vogel-Heuser, C. Diedrich, A. Fay, S. Jeschke, S. Kowalewski, M. Wollschlaeger, and P. Göhner. Challenges for software engineering in automation. *Software Eng. and Applications*, 7(5):440, 2014.

[10] L. Guang, G. Plosila, J. Isoaho, and H. Tenhunen. Hamsoc. In *Autonomic Networking-on-Chip: Bio-Inspired Specification, Development, and Verification*, pages 135–164. CRC Press, 2011.

[11] L. Guang, J. Plosila, J. Isoaho, and H. Tenhunen. Hierarchical agent monitored parallel on-chip system: A novel design paradigm and. *Innovations in Embedded and Real-Time Systems Engineering for Communication*, page 278, 2012.

[12] S. Sarma, N. Dutt, P. Gupta, A. Nicolau, and N. Venkatasubramanian. On-chip self-awareness using cyberphysical-systems-on-chip (cpsoc). In *Proc. of the 2014 CODES*, page 22. ACM, 2014.

[13] S. Sarma and N. Dutt. Fpga emulation and prototyping of a cyberphysical-system-on-chip (cpsoc). In *Rapid System Prototyping (RSP), 2014 25th IEEE Int. Symp. on*, pages 121–127. IEEE, 2014.

[14] H. Hoffmann, M. Maggio, M. D. Santambrogio, A. Leva, and A. Agarwal. Seec: A framework for self-aware computing. 2010.

[15] L. Motus, M. Meriste, and J. Preden. Towards middleware based situation awareness. In *MILCOM*, pages 1–7. IEEE, 2009.

[16] J. Preden, J.s Llinas, G. Rogova, R. Pahtma, and L. Motus. On-line data validation in distributed data fusion. In *SPIE Defense, Security, and Sensing*, page 87420S. Int. Soc. for Optics and Photonics, 2013.

[17] D. Vernon, G. Metta, and G. Sandini. A survey of artificial cognitive systems: Implications for the autonomous development of mental capabilities in computational agents. *IEEE Trans. on Evol. Computation*, 11(2):151–180, 2007.

[18] A. Wendt and T. Sauter. Agent-based cognitive architecture framework implementation of complex systems within a multi-agent framework. In *ETFA, 2016 IEEE 21st Int. Conf.*, pages 1–4, 2016.

[19] J. E. Laird. *The Soar cognitive architecture*. MIT Press, 2012.

[20] S. Gottifredi, M. Tucat, D. Corbatta, A. J. García, and G. R. Simari. A bdi architecture for high level robot deliberation. In *XIV Congreso Argentino de Ciencias de la Computación*, 2008.

[21] N. Trivedi, P. Langley, P. W. Schermerhorn, and M. Scheutz. Communicating, interpreting, and executing high-level instructions for human-robot interaction. In *AAAI Fall Symp.: Advances in Cog. Systems*, 2011.

[22] J. Van Oijen, W. Van Doesburg, and F. Dignum. *Goal-based communication using bdi agents as virtual humans in training: An ontology driven dialogue system*. Springer, 2011.

[23] S. Franklin, S. Strain, R. McCall, and B. Baars. Conceptual commitments of the lida model of cognition. *Journal of AGI*, 4(2):1–22, 2013.

[24] S. Schaat, A. Wendt, S. Kollmann, F. Gelbard, and M. Jakubec. Interdisciplinary development and evaluation of cognitive architectures exemplified with the sima approach. In *EAPCogSci*, 2015.

[25] G. Zucker, A. Wendt, L. Siafara, and S. Schaat. A cognitive architecture for building automation. In *Industrial Electronics Society, IECON 2016-42nd Annual Conf. of the IEEE*, pages 6919–6924. IEEE, 2016.

[26] A. Bratukhin and T. Sauter. Distribution of mes functionalities for flexible automation. In *2010 IEEE Int. Workshop on Factory Com. Systems Proc.*, pages 157–160, 2010.

[27] A. Bratukhin, A. Nagy, and A. Mahmood. Distribution of control functionality in energy-aware industrial building environment. In *Factory Com. Systems (WFCS), 2015 IEEE World Conf.*, pages 1–8. IEEE, 2015.

[28] J. Wang and A. Kusiak. *Computational Intelligence in Manufacturing Handbook*. CRC Press, 2000.

[29] A. Luder, A. Klostermeyer, J. Peschke, A. Bratoukhin, and T. Sauter. Distributed automation: Pabadis versus hms. *IEEE Trans. on Ind. Informatics*, 1(1):31–38, 2005.

[30] H. Kuehnle, U. Bergmann, and A. Lder. Design of multi-agent decision support for configurations of manufacturing networks. In *2009 IEEE ICE*, pages 1–8, 2009.

[31] R. F. Babiceanu and F. F. Chen. Development and applications of holonic manufacturing systems: a survey. *Journal of Intelligent Manufacturing*, 17(1):111–131, 2006.

[32] A. A. Abbasi and M. Younis. A survey on clustering algorithms for wireless sensor networks. *Comp. Com.*, 30(14):2826–2841, 2007.

[33] X. Liu. A survey on clustering routing protocols in wireless sensor networks. *sensors*, 12(8):11113–11153, 2012.

[34] M. Saleem, G. A. Di Caro, and M. Farooq. Swarm intelligence based routing protocol for wireless sensor networks: Survey and future directions. *Information Sciences*, 181(20):4597–4624, 2011.

[35] P. J. Marron, S. Karnouskos, D. Minder, and A. Ollero. *The emerging domain of Cooperating Objects*. Springer Science & Bus. Media, 2011.

[36] N. TaheriNejad, A. Jantsch, and D. Pollreisz. Comprehensive observation and its role in self-awareness; an emotion recognition system example. In *FedCSIS*, pages 123–130, 2016.

[37] M. Götzinger, N. TaheriNejad, H. A. Kholerdi, and A. Jantsch. On the design of context-aware health monitoring without a priori knowledge; an AC-motor case-study. In *Can. Conf. on Elec. and Comp. Eng.*, 2017.