Integration Patterns for Interfacing Software Agents with Industrial Automation Systems

Paulo Leitão*, Stamatis Karnouskos†, Luis Ribeiro‡, Panayiotis Moutis§, José Barbosa*, Thomas. I. Strasser¶

*Research Centre in Digitalization and Intelligent Robotics (CeDRI), Instituto Politécnico de Bragança, Campus de Santa Apolónia, 5300-253 Bragança, Portugal, email: {pleitao, jbarbosa}@ipb.pt
†SAP, Walldorf, Germany, email: stamatis.karnouskos@sap.com
‡Linköping University, SE-58 183 Linköping, Sweden, email: luis.ribeiro@liu.se
§Carnegie Mellon University, Pittsburgh, PA, United States, email: pmoutis@andrew.cmu.edu
¶Center for Energy – AIT Austrian Institute of Technology and Institute of Mechanics and Mechatronics – Vienna University of Technology, Vienna, Austria, email: thomas.i.strasser@ieee.org

***Abstract*—Agent-based systems, an approach derived from distributed artificial intelligence, have been introduced for designing large complex systems. They are also suitable to solve challenging problems in industrial environments, being an appropriate technology for realizing cyber-physical systems. In such configuration, they need to interface with automation and control devices. However, until now there is no widely accepted practice nor pattern to interface the software agents with the automation functions. This work addresses this issue and introduces corresponding integration patterns in order to achieve full interoperability and reusability. This work, therefore, provides a methodology for mapping existing practices into a set of generic templates and also discusses the applicability of the proposed approach to different industrial application domains.**

I. Introduction

Multi-Agent Systems (MAS) is a technology, derived from the distributed artificial intelligence, that provides an alternative way to design complex large-scale systems by decentralizing the intelligence, adaptation, and control by distributed, autonomous and cooperative entities [1]. The MAS concept differs from conventional automation and control approaches due to its inherent capabilities to adapt to emergence without external intervention.

Industrial agent-based solutions are suitable to realize Cyber-Physical Systems (CPS), aligned with Industry 4.0 principles, expanding the potential application domains of MAS through the distribution of intelligence and at the same time adding the required flexibility, robustness and responsiveness to industrial automation systems [2]–[4]. Industrial agents are naturally facing several industrial requirements, namely specific hardware integration, reliability, fault-tolerance, scalability, industrial standard compliance, quality assurance, resilience, manageability and maintainability [3], [5].

Particularly, one of the major requirements of industrial agents is their capability to interface low-level control of industrial automation systems [3], [4], [6]. As examples, consider one software agent associated to a punching machine and required to interact with a Programmable Logic Controller (PLC) controlling the automation system to write/read variables, or a software agent associated to a smart metering device to gather the current consumption data.

Today, there is no widely accepted (not to mention standardized) practice on how to interface software agents and physical hardware. Several approaches have flourished in practice [7], but most of them are either on the concept level and typically based on proprietary technologies or custom-defined interfaces are being used. Additionally, for the existing empirical practices, there is hardly a way to compare them due to their variety, as well as vice versa it is difficult to propose a best practice to follow during the design and engineering phase of a specific use-case.

In order to achieve full interoperability, the pertinent question that arises is how to interface in an easy, transparent, and reusable way the software agents to the different automation systems and devices. Since each automation device controller has a proper, usually proprietary, interface, it is necessary a standardized interfacing approach to make this a reality.

The IEEE P2660.1 Working Group [8] is developing approaches to derive a recommended practice for the integration of software agents and industrial automation systems, in specific scenarios. The life-cycle of this work, illustrated in Figure 1, comprises several phases, namely:

- *Analysis phase*, related to the collection and analysis of existing interface practices such as for the factory automation, power and energy systems and building automation application domains [7].
- *Generalization phase*, related to the definition and characterization of generic interface templates derived from the previous phase.
- *Assessment phase*, related to the definition of an assessment methodology and including the definition of a test plan and the execution of the assessment of key characteristics for the identified interface templates.
- *Recommendation phase*, in order to derive the recommended practices for a specific set of requirements and taking into consideration the results from the previous assessment phase.

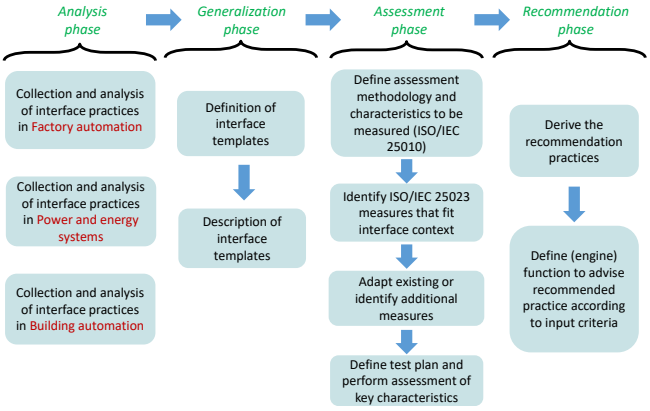

Figure 1. Life-cycle of P2660.1 working development.

This work focuses on the generalization phase and particularly the description of the generic interface practices (templates) for interconnecting software agents and low-level automation functions, starting for the methodology to classify these generic practices and then formally characterize them according to the logical, process, functional, and physical views. Other phases such as the assessment methodology are discussed in other work [9].

The remaining part of this paper is structured as follows: section II outlines the related work and section III describes the methodology to map the existing interface practices in a set of templates. section IV characterizes each template according to logical, process, functional and physical views, and section V discusses the applicability of these generic interface practices to the different application domains. Finally, section VI rounds up the paper with the conclusions.

II. Current Practices and Related Work

A survey study performed under the IEEE P2660.1 Working Group [8] activities has uncovered a wide range of different integration practices and patterns across three application domains: *(i)* factory automation, *(ii)* power and energy systems, and *(iii)* building automation [7]. The survey practices confirmed the prevalence of the two-layered approach and exposed a broad set of interaction and communication protocols as well as High-Level Control (HLC) integration practices [3], [6]. More particularly, the HLC has been deployed within and outside the automation controller and its interaction with the Low-Level Control (LLC) has ranged anywhere from almost direct control through shared memory spaces to brokered message based-interaction.

Other significant conclusions from this survey are that Java is the primary programming language to codify the agents (probably due to the wide utilization of the JADE [10]), followed by C++, and that security is usually not considered in such applications. On the other side widespread approaches for the realization of automation functions are from PLC domain; i.e., domain standards IEC 61131 and IEC 61499 are quite often used [3], [6], [11]. Usually, the interfaces follow a kind of proprietary protocols, but recently the use of Machine-to-Machine (M2M) communication technologies such as Open Platform Communications – Unified Architecture (OPC-UA) or Message Queuing Telemetry Transport (MQTT), is being noticed, increasing the transparency and industrial adoption [12]. The reported interfaces usually follow a coupled approach (i.e., agents are running remotely from the automation controller) and a client/server interaction mode, and providing read/write operations on the digital I/O interaction as main functionality.

Furthermore, for the majority of the cases, software agents are applied within monitoring, control and simulation domains, where security aspects are not currently seen as a cornerstone at the design phase.

III. Methodology for Mapping Existing Practices

The analysis of surveyed practices allowed to identify the similarities between them, as described in the previous section. Aiming to establish the generic interface practices, the documented interface practices can be classified as illustrated in Figure 2.

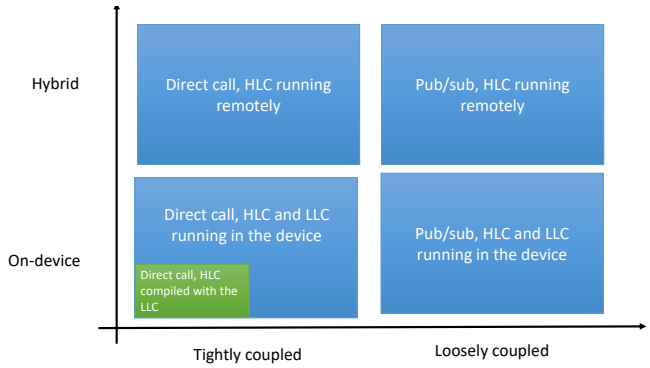

Figure 2. Mapping the generic interface practices among interaction mode and location levels of abstraction.

Specifically there are two distinct abstraction levels:

- *Interaction Mode (L0)*, related to the interaction mode between parties, i.e., the agent and the LLC. Several different possibilities:
  - *Tightly Coupled*, where there is a direct, permanent and non-mediated connection (sync) between the two parties (i.e., HLC and LLC), e.g., via Remote Procedure Call (RPC). Such connection can exist in the form of direct network communication or shared memory, and the interaction protocol follows the traditional request-response schema, as illustrated in Figure 3. In this case, the HLC can exert more direct control over the LLC.
  - *Loosely Coupled*, where there is a mediated connection between parties (async), e.g., via a queue or a pub/sub channel. In this case, the interaction is not direct and is rather mediated typically by one or several message brokers, as illustrated in Figure 4.
- *Location (L1)*, related to the location of the parties, i.e., where the agent and LLC are hosted. Different

Preprint version of doi:10.1109/IECON.2018.8591641

possibilities (considering that LLC is always within the device) should be considered:

- *Hybrid*, where the agent is running externally to the device and the LLC is on-device, i.e., they are running in different computational platforms.
- *On-device*, where both the software agent and LLC are co-hosted in the device, i.e., they are sharing the same computational platform.

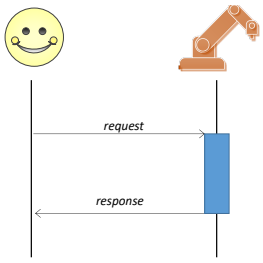

Figure 3. Interaction protocol for the *Tightly Coupled* interface.

Considering the classification based on the tuple *{interaction mode, location}*, four major interface practices can be derived (see Figure 2): (i) {Tightly Coupled, Hybrid} (ii) {Tightly Coupled, On-device} (iii) {Loosely Coupled, Hybrid} (iv) {Loosely Coupled, On-device}.

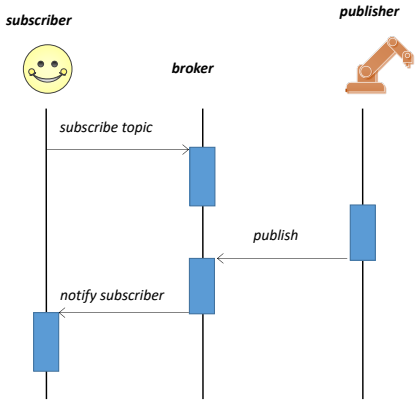

Figure 4. Interaction protocol for the *Loosely Coupled* interface.

These vary on the location of the software agent or HLC and the LLC as well as on their interface. In this context, the HLC and the LLC can share the same computational platform (On Device) or can be in different ones (Hybrid). The HLC can exert more direct control on the LLC (Tightly coupled) or the interaction can be mediated by a broker (Loosely Coupled). There is also one particular case of a tightly coupled and on device design which results from the HLC and the LLC being compiled and deployed as a single set of binaries.

IV. Characterization of Interfacing Practices

This section characterizes the identified generic interface practices, describing the logical, process, functional, and physical views. In this document, the different views have the following meaning:

- *Logical View*, which describes the system objects and their inter-connections, providing a conceptual perspective of the practice.
- *Functional View*, which is usually an elaboration of the logical view (so it can be considered as part of the logical view itself), providing the functionality that the system provides to end-users.
- *Process View*, which deals with the dynamic aspects of the interface practice, explaining how the system processes communicate, and focusing on the runtime behavior. This view addresses, e.g., concurrency, distribution, performance and scalability.
- *Physical View*, which discusses the system from a system engineer's point of view, addressing the topology of software components on the physical layer as well as the physical connections between these components (assignment of code to non-software resources such as hardware, as well as the used technologies).

The next sub-sections describe the four interface templates according to these perspectives.

A. Interfacing Practice: Tightly Coupled, Hybrid

This interfacing practice is characterized by the schema where the software agents (HLC) are running remotely and are accessing, using a direct connection, the physical automation device, which is running a native logic control layer that ensures (real-time) responsiveness, to get data or send commands (LLC). In this pattern, there is a computational separation of the HLC and LLC, as illustrated in Figure 5, where both ends will implement communication mechanisms that enable them to interact. Usually, the LLC Application Programming Interface (API) exposes a local data memory (read/write) table and/or by use of a function/service-based API to support the exchange of data and/or the invocation of functions between the HLC and LLC ends, where the HLC-side accesses the LLC functionalities by means of a communication channel. The channel can be implemented as a socket-based approach or using other network-based direct call communication channels. Naturally, this pattern is affected by the quality of the channel that may lie in-between, and the computational capabilities at both ends or even due to a parallel process being executed at either side. The network traffic and the increase number of industrial agents in the system will also contribute to the degradation of the pattern quality, namely concerning response time.

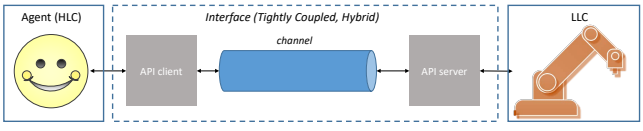

Figure 5. Interface structure for the *{Tightly Coupled, Hybrid}*.

Regarding the process view, the HLC is accessing the LLC using a direct connection, following a request-response protocol, as illustrated in Figure 3. This interaction pattern

Preprint version of doi:10.1109/IECON.2018.8591641

is used to perform several types of requests, namely read and write of variables (e.g., I/Os), download, upload, start and stop programs, and subscribe events.

B. Interfacing Practice: Tightly Coupled, On-device

This interfacing practice is characterized by a schema where the software agents are directly embedded in the automation devices (see Figure 6). More precisely this means that the automation device includes computational support for the agent binaries and that such support allows native access to the functions of the device. This access may be granted for soft or hard real-time operations but the agent binaries and native control layer share the same physical container. The electronics within the automation device ultimately define the concrete integration procedure (data bus, internal network connection, shared memory, etc.). However, from the software point of view, the integration will be mediated by a native software library. Such library may make internal network connections or buses more or less explicit but ensure the integration from a software point of view. Such library can be loaded dynamically at runtime as part of some agent initialization routines or it can be compiled together with the agent binaries. In the very extreme case, the agent itself is compiled together and included in the native software of the controller.

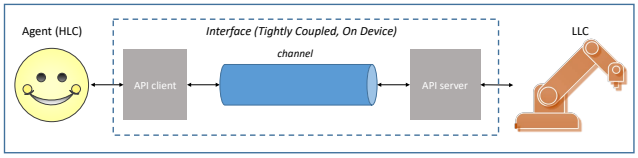

Figure 6. Interface structure for the {Tightly Coupled, On-device}.

Embedding the HLC into the LLC mitigates, in theory, the network related issues, particularly latency, and should improve the overall performance if the device has enough computational power to support both the HLC and the LLC while fulfilling the timing requirements of both.

Regarding the process view, this interface practice follows the same schema as exhibited by the {Tightly Coupled, Hybrid}, i.e., the HLC is accessing the LLC using a direct connection, as illustrated in Figure 3. This interaction pattern allows to perform several types of requests, namely read and write of variables (e.g., I/Os), download, upload, start and stop programs, and subscribe events.

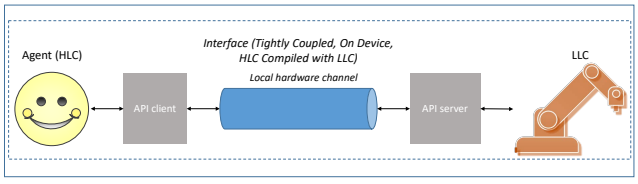

Figure 7. Interface structure for the {Tightly Coupled, On-device}, considering the case that HLC is jointly compiled with LLC.

As briefly mentioned before, two particular cases can be identified in this interface practice:

- The sharing of the controller computational resources through a software library exposing the different native control functions but abstracting from a development point of view the concrete electronic implementation. In this case, the developer may know that it is using particular electronics but the details are hidden by the library. This will be the usual case as it keeps Intellectual Property (IP) protection over commercial hardware implementations. However such practice will generally restrict the performance to soft real-time control or otherwise to hard-real-time control on a higher time scale.
- The rather unusual case whereby the agent code is compiled together with the controller's native binaries (see Figure 7). Such practice can provide hard real-time assurances, but there is currently a lack of proper tools for developing the agent-based part of the practice.

C. Interfacing Practice: Loosely Coupled, Hybrid

The most typical design related to the loosely coupled schema includes the computational separation of the HLC and the LLC in terms of location where they are running, as shown in Figure 8. Typically, this interface uses a third-party entity between the HLC and the LLC. The most common topology uses a broker-mediated channel following a publish/subscribe approach where the HLC subscribes to events provided/originated by the physical controller (i.e., the device that is physically connected to the process under control and/or being monitoring). On the other side, the LLC subscribes to the control topics, published from the HLC, that are needed for a proper process control. Several technological protocols are being currently widely industry accepted, namely OPC-UA, MQTT or XMPP.

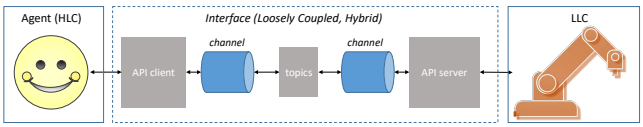

Figure 8. Interface structure for the {Loosely Coupled, Hybrid}.

As with the hybrid coupled design, the performance of the system depends on the capabilities of the network infrastructure and in this case of the message brokering systems as well. This implies that for highly intensive communication scenarios or for hard-real-time situations, this approach might have limited approaches. While for the first situation, a load-balancing strategy at the broker level might ease the issue, on the second, i.e., in a hard-real-time situation, the loose couple approach besides having the network latency it still has the broker logical process-flow to delay the message exchange between the HLC and the LLC.

On the other side, for very large systems, this approach may scale better than coupled designs. Furthermore, by promoting this loosely coupled approach and being the HLC running, normally, in high computational hardware, complex algorithms might be added at the HLC side without ever interfering with the normal operation at the LLC level. Other functionality

Preprint version of doi:10.1109/IECON.2018.8591641

could be seen in situations where an HLC could/need to publish topics to a multitude of LLC devices. It is worthy to note that in such situations this approach could offer a great design advantage. Similarly, a loosely coupled hybrid approach could be advantageous in situations where more than one HLC could/need to issue control/monitoring commands to the same LLC (note that in this paper the authors are not evaluating the increase of the design complexity of such approach).

Additionally, manufacturers are widely offering publish/subscribe interface mechanisms, e.g., OPC-UA, MQTT or XMPP, making this interface catch increasingly the attention and developments, both from manufacturers but also from system integrators in the years to come.

D. Interfacing Practice: Loosely Coupled, On-device

The schema of this practice, as shown in Figure 9, has the software agent embedded to the automation device and the HLC and LLC interact through a brokered interface. The practice describes a computation separation between the agent and the automation device, although they are integrated into the same physical unit.

From the functional point of view, the message broker is the compulsory intermediate between the HLC and the LLC. This means that the HLC and the LLC do not exchange direct messages of any shorts, but go through the broker as presented in Figure 4. The common placement of both the agent and the automation device implies that the effect of the broker in the interface performance is limited to concerns of just one hardware framework.

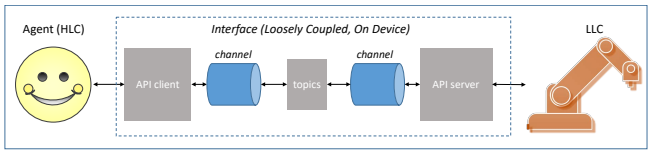

Figure 9. Interface structure for the {Loosely Coupled, On Device}

Thanks to the broker interface, supposedly offering a superset of functionalities (compared to any number of functionalities represented by either the HLC or the LLC), the user may implement any number and type of control actions (I/O, start/stop programs, etc.) on any subset of devices. This means that some devices may have some control capabilities, while other devices different ones. Thus, the practice offers privacy, flexibility, control/functional grouping, although it burdens the user with the definition of the control capabilities. The coexistence of the agent and the automation device on the same piece of hardware reduces the complexity of processing the said definitions since they can be implemented at once for both ends of this practice. The broker interface is counterintuitive to the embedded integration of the agent in the control device hardware since it implies an extra layer for realizing any control action. A realistic case where such a practice may be inevitable is when an automation device with specific hardware characteristics is enhanced with agent-hosting capabilities and vice versa. In such a case the role of a broker facilitates an easier transition than defining direct (tightly coupled) native control automation access for the agent. The user is also burdened with properly updating all HLC-LLC definitions across all hardware, whenever the broker is updated with functionalities that are required to be added to the interface.

As from the purely physical aspect of this practice, either end of this topology may be exclusively developed based on the broker (most probably the automation device) or the broker is a shared ontology/vocabulary for both the HLC and the LLC. Evidently, the loose coupling means that the agent cannot directly control or monitor any part of the automation device, i.e., the agent and the automation software reside on different processing blocks of the same piece of hardware. The latest remark implies that the practice is fit when customization is required (from many aspects) for specific hardware platforms but is industry-friendlier thanks to the broker interface that may be broadly and deeply standardized.

The performance of this design depends highly on the broker, while it is also not an intuitive architecture for brokered interactions (unlike for example when considering hybrid topologies, with the HLC and the LLC on different hardware units).

V. Discussion

As observed there are several integration patterns, when it comes to software agents and industrial low-level automation functions as presented in section IV. For the software part, all of these are a subset of well-known integration patterns [13]. A sampling of some of the agent relevant implementations [7] reveals that only some of them are evident in production. As there is a lack of any comparative analysis among them, nor any knowledge repository for design and in-use experiences, the only source of information seems to be practical experience and some literature in the domain. Therefore, a decision on what practice to use seems to be constrained to its technical aspects only, which is partially justified, as also other aspects ought to be also considered [5].

The potential directions as illustrated in Figure 2, provide hints to potential strategies that could be followed. In industrial settings, the case of brownfield landscapes is usually the case, which implies that any solution proposed has to be integrated with legacy (i.e, already existing and deployed) systems. The constraints set, sometimes apply to the preservation of the physical infrastructure, while software changes are permitted. The latter implies for instance that no new hardware should be installed (or it should be “embedded” within existing infrastructure, e.g., a PLC rack). Such physical constraints set the criterion of exclusion of some approaches, e.g., the co-location of a new industrial PC, where the agent platform could be running and is required for some hybrid scenarios. Similarly, if the existing device, e.g., the PLC, cannot accommodate the requirements of an agent system, and no additional hardware positioning is possible, the implementation of agents in a hybrid form, where the agent platform is for example in the cloud may be the only solution. In fact, this situation where a multi-agent system need/could be designed for a

Preprint version of doi:10.1109/IECON.2018.8591641

given scenario might be one of the most promising solutions where a set of data repositories are installed throughout the shop-floor enabling the connection of the HLC with the LLC data. Normally, these are polling the LLC using the available communication protocols and are afterward exposing this information in more transparent and widely common formats e.g., via the information model of OPC-UA.

The physical and software implementation requirements set some context, upon which some practices may be excluded. The specific domain constraints (e.g., performance), impose additional considerations. For instance, a publish/subscribe approach where the agent is located in the cloud, and has to communicate via a best effort network, may not be seen as fit for real-time scenarios, as the network latency and communication delays cannot be guaranteed.

Even if some of the potential practices remain as valid selections, their realization may highly differ. Here, one has to think not only about the short term, but also a potential infrastructure evolution (long-term) strategy. It may be seen as highly efficient to have an implementation on-device with a direct call in a certain set-up. At the same time, in other cases, integrating multiple different types of LLC in a given infrastructure might hint towards a preference of all LLC types sharing a common broker messaging system, so that the HLC need not be disrupted as the infrastructure evolves. However, in general, such implementations are usually highly customized, and are difficult to change in order to accommodate future needs that arise from the system usage and needs e.g. for security updates to the system.

As already discussed, a selection of a practice is not seen as straightforward, and is a more complex issue that goes beyond technical only aspects [5]. The variety of the existing practices attests the lack of a one-size-fits-all solution, and best-practices depend on specific use cases, as well as domain requirements.

Additionally, the overwhelming majority of the experiences with software agents and low-level automation system integration is via traditional dedicated industrial wired systems. However, the introduction of new wireless technologies and especially the low-power ones [14] that may even operate over lossy networks is not sufficiently addressed and needs to be assessed.

VI. Conclusions

In the context of Cyber-Physical Systems, the implementations of industrial agents face an important need regarding the interface with automation and control devices [2], [3]. However, at the moment, there is no widely accepted practice nor pattern to interface the agents with the automation functions. The P2660.1 Working Group [8] established by the IEEE Standards Association, aims to develop recommended practices that provide full interoperability and reusability to solve this problem.

This work focuses on the generalization of the interface practices, derived from the compilation and analysis of existing interface practices such as for the factory automation, power and energy systems and building automation application domains. The proposed four interface templates are classified according to the tuple *{interaction mode, location}*, where the interaction mode refers the way HLC and LLC interact (i.e., Tightly Coupled or Loosely Coupled), and the location refers the location where HLC and LLC are hosted (i.e., Hybrid and On-device). The characterization of each interface template was performed according to the logical, functional, process and physical perspectives. The paper also discusses the applicability of the interface templates, particularly analyzing the pros and cons of each one to address industrial applications.

As a general conclusion, it is clearly noticed that there isn't a generic interface that fits the whole spectrum of applications. Furthermore, the selection of one of the four identified practices is not straightforward and is intrinsically connected not only to the application domain but also to the existing legacy systems and the long-term vision that is followed by the company.

Future work will be devoted to the remaining phases of the life-cycle methodology, and particularly the assessment of each interface template according to a proper set of characteristics that will allow deriving the recommended practices.

References

[1] M. Wooldridge, *An Introduction to Multi-Agent Systems*. John Wiley and Sons, 2002.

[2] P. Leitão, S. Karnouskos, L. Ribeiro, J. Lee, T. Strasser, and A. W. Colombo, "Smart agents in industrial cyber–physical systems," *Proceedings of the IEEE*, vol. 104, no. 5, pp. 1086–1101, May 2016.

[3] P. Leitão and S. Karnouskos, *Industrial Agents: Emerging Applications of Software Agents in Industry*. Elsevier, 2015.

[4] V. Marik and D. McFarlane, "Industrial adoption of agent-based technologies," *IEEE Intelligent Systems*, vol. 20, no. 1, pp. 27–35, Jan. 2005.

[5] S. Karnouskos and P. Leitão, "Key contributing factors to the acceptance of agents in industrial environments," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 2, pp. 696–703, Apr. 2017.

[6] P. Vrba, P. Tichý, V. Mařík, K. H. Hall, R. J. Staron, F. P. Maturana, and P. Kadera, "Rockwell automation's holonic and multiagent control systems compendium," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 41, no. 1, pp. 14–30, Jan. 2011.

[7] P. Leitão, S. Karnouskos, L. Ribeiro, P. Moutis, J. Barbosa, and T. I. Strasser, "Common practices for integrating industrial agents and low level automation functions," in *IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society*. IEEE, Oct. 2017.

[8] IEEE Project P2660.1 – Recommended Practices on Industrial Agents: Integration of Software Agents and Low Level Automation Functions. [Online]. Available: https://standards.ieee.org/develop/project/2660.1.html

[9] S. Karnouskos, R. Sinha, P. Leitão, L. Ribeiro, and T. Strasser, "Assessing the integration of software agents and industrial automation systems with ISO/IEC 25010," in *IEEE International Conference on Industrial Informatics (INDIN)*, 2018.

[10] F. L. Bellifemine, G. Caire, and D. Greenwood, *Developing multi-agent systems with JADE*. Chichester, England Hoboken, NJ: John Wiley, 2007.

[11] G. Zhabelova and V. Vyatkin, "Multiagent smart grid automation architecture based on iec 61850/61499 intelligent logical nodes," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 5, pp. 2351–2362, 2012.

[12] W. Mahnke, S.-H. Leitner, and M. Damm, *OPC unified architecture*. Springer Science & Business Media, 2009.

[13] G. Hohpe, *Enterprise integration patterns : designing, building, and deploying messaging solutions*. Boston: Addison-Wesley, 2004.

[14] T. Watteyne, V. Handziski, X. Vilajosana, S. Duquennoy, O. Hahm, E. Baccelli, and A. Wolisz, "Industrial wireless IP-based cyber –physical systems," *Proceedings of the IEEE*, vol. 104, no. 5, pp. 1025–1038, May 2016.

Preprint version of doi:10.1109/IECON.2018.8591641