Andrea Bonci*, Massimiliano Pirani and Sauro Longhi

A Database-Centric Framework for the Modeling, Simulation, and Control of Cyber-Physical Systems in the Factory of the Future

DOI 10.1515/jisys-2016-0281

Received October 30, 2016; previously published online July 14, 2017.

Abstract: The factory of the future scenario asks for new approaches to cope with the incoming challenges and complexity of cyber-physical systems. The role of database management systems is becoming central for control and automation technology in this new industrial scenario. This article proposes database-centric technology and architecture that aims to seamlessly integrate networking, artificial intelligence, and real-time control issues into a unified model of computing. The proposed methodology is also viable for the development of a framework that features simulation and rapid prototyping tools for smart and advanced industrial automation. The full expression of the potentialities in the presented approach is expected in particular for applications where tiny and distributed embedded devices collaborate to a shared computing task of relevant complexity.

Keywords: Database management systems, distributed computer control systems, embedded systems, industry automation, intelligent manufacturing systems, simulators.

2010 Mathematics Subject Classification: 68, 93, 94.

1 Introduction

The “factory of the future”, “smart manufacturing”, “Industry 4.0”, or “the digital enterprise” are overlapping names of the strategic initiatives in technologies for the expected and incoming revolution of the industrial field. Cyber-physical systems (CPS) are involved in this new momentum, as they make promises about some effective handling of the complexity of the goals in industrial research. CPS make treatable the contexts where physical and software components are deeply intertwined to interacting with each other in a myriad of ways that change with context [26]. The efficiency measurements and assessment of industrial production processes in the future scenario have to take into account the increasing role of information flow across processes, from the enterprise system level down to the shop-floor. Information is the key. Through information control, transmission, and treatment, humans expect to be in better control of sustainability and productivity performance. Furthermore, the introduction of artificial intelligence (AI) as a decision support tool and some collaborative swarms of automated agents is a hot theme as well for the control of complex systems of systems [40].

The appearance of unforeseen behaviors is a typical phenomenon in complex systems. New behaviors, features, and prospects might emerge from the deliberate application of data mining techniques coupled with artificial reasoning and inference, still on already well-known and established data. New opportunities would originate from the full exploitation of information acquired, stored, and communicated in industrial processes.

*Corresponding author: **Andrea Bonci**, Department of Information Engineering (DII), Università Politecnica delle Marche, via Brecce Bianche, 60131 Ancona, Italy, e-mail: a.bonci@univpm.it. <http://orcid.org/0000-0003-0265-1598>

Massimiliano Pirani and Sauro Longhi: Department of Information Engineering (DII), Università Politecnica delle Marche, via Brecce Bianche, Ancona, Italy

This paper proposes an information-conservative approach suggesting a key enabling technology and a methodology that is suitable for the realization of a lightweight framework for the modeling, control, simulation, planning, optimization, and scheduling of industrial processes. The key approach is the pervasive use of relational database systems that actively support the transmission, storage, and elaboration of information across the five levels defined in the ISA-95 standard or across all the components of the new reference architecture RAMI 4.0 [42] – from sensing and actuation to the management of a network of enterprises across the whole life-cycle and sustainability issues. In the future, it is expected to move from the existing hierarchical control structures, based on the ISA-95 automation pyramid, toward more decentralized and reconfigurable structures based on the CPS principles [28]. Indeed, cyber-physical production systems (CPPS) will show cyber-capabilities within every physical component as distributed computing along with distributed intelligence and self-methods, namely, self-adaptation and self-configuration, along with self-diagnosis, self-organization, and self-reconfiguration dynamics.

The adoption of an active distributed database mechanism at the shop-floor can be shown as feasible and can be realized through the best embedded database technologies today available [35]. This technique is candidate to render the data mining and the optimization of processes viable also in the presence of complex CPS problems.

By the use of the quality of a declarative language, as are the database languages, most of the techniques for planning and optimization [23] can be enabled dynamically and with reduced effort both from humans and from the machines. Decision support systems based on time-aware relational model (RM) inference can lead toward results potentially unforeseen at the beginning of the information gathering [13, 31, 43]. The full RM is still not available technology and deserves more scientific effort [13], at least not widely adopted in commercial solutions and still not available for its effective deployment into tiny embedded devices. Nevertheless, a restricted database-centric technology based on the established SQL database language standard can create at least the first technological step toward the challenges created by the smart manufacturing scenario.

Here, we propose a research and technological path toward the simplification and harmonization of the existing technologies basing on a rather simple technique. Each solution, both for communication and the computation in CPS, features some structured data at its core. This data core can be the common ground for effective harmonization of technologies. To cope with elaboration and communication of structured data, semantics database technology and models are, to the author's knowledge, the best and more natural approach. Some experience in the field [7, 8] and new commercial trends [22] have started to testify that paying more attention to database technology at the core can at least suggest to researchers and practitioners a promising path for CPS. A focus on the capabilities and implementation possibilities of the RM, in Codd's sense [10] and its real-time semantic possibilities [13], are worthy of investigation. The focus on database management systems (DBMS) must be a key for the unification and the sanitizing of the lack of interoperability and semantics in software and hardware tools and languages. The design of the unifying technology to be pursued should take into account the following prescriptions:

- (i) All the information needed by the control, configuration, and logic is inside the relational database.
- (ii) All the components of the CPS should use the RM as their unified and unique interface.
- (iii) The human and artificial agents, both as operators and programmers, have a shared data and information model to comply with.
- (iv) A clear separation between the declarative part, used for deliberate learning, planning, and dynamical programming for system intelligence, and the procedural part involving real-time and implementation efficiency. These two parts should be compliant with the same data model.

With the use of active and distributed embedded database, we propose a framework that dynamically allows the components, as plug-ins, to participate in the system. A component using a set of queries can know what the system needs and what role is the most suitable to participate and can then configure and program itself accordingly. In this view, database languages are already expressive enough and can be interfaced easily with any other languages. This impinges with the AI possibilities and related multiagent

system technologies. There are two levels to keep in mind for efficient implementations: deliberate learning and adaptation (declarative step) and then metaprogramming of procedures (procedural step). The present work proposes guidelines and technology hints that can be effectively used in KPI-based control for the sustainability and efficiency of industrial processes within the sustainable factory of the future research framework [39].

In Section 2, we introduce the related work on the basic ideas of the database-centric technology. In Section 3, a problem description along with a technology description and its unifying role for the industry is provided. In Section 4, a simulation and modeling methodology is introduced. In Section 5, the results expected from the methodology are discussed. Section 6 provides a discussion on the positioning of the proposed methodology with respect to recent advances in the field. Section 7 is dedicated to conclusions.

2 Pervasive Database Technology and Related Work

Industrial production processes are complex, and the heterogeneity of the components and protocols across the factory floor, and its organizational and geographical boundaries, creates strong demands for system integration and interoperability [20]. Valuable unifying efforts toward standards, such as the OPC UA [30], are well established. In any case, the OPC UA computing model is not designed to be able to scale down to the lowest class of devices in the future shop-floor and keep up with the incoming CPS storm mainly because those issues were not present at the time of its design. Whereas in OPC UA the database is used mainly for historical data storage, in the following proposed approach, each component or agent in the process is provided with active database capabilities that tracks data and events to trigger control on the plant. Information is made constantly available for its use at the highest strategic levels, where decisions cope with the long-term issues. Decisions made at the strategic level can influence the procedures implemented at lower levels through dynamical programming based on planning and reasoning with context awareness. Using distributed computing techniques and tools, we can keep the costs of complex solutions low and viable. The technique consists of adding a database everywhere, which does not necessarily require a change at the lower-level technology or acquisition system. It can be applied on top of existing facilities.

Distributed databases for CPS imply networking, as we have to include the Internet of Things (IoT) inside. In Ref. [24], it is shown how the IoT promise of integrating the digital world of the Internet with the physical world needs to be implemented through a systematic approach for integrating the sensors, actuators, where data are the central entity for the realization of context-aware services. A complete review about how IoT encompasses smart objects can be found in Ref. [16], whereas the relevant security issues arising within the IoT context are analyzed and classified in Ref. [15]. In Ref. [36], it is shown how the abstraction of models is necessary in the current complexity of CPS. For this kind of abstract modeling, we put forth the expressiveness of the RM or, where not fully available, at least the ordinary relational database technology with the plethora of extensions provided from SQL database system vendors.

Nowadays, some industrial producers such as Inductive Automation™ already use this kind of database-centric technique [22]. They use an SQL database as the center of every software and business logic of their industrial automation application. In Ref. [14], the authors showed how data processing can be integrated and performed within the DBMS. Both the formerly cited solutions are going in the right direction lest the necessary scalability foreseen in CPS is still lacking.

In CPS, the database must scale down to use a minimum of resources (potentially below 1 MB of memory). This is needed increasingly more, as most embedded devices will belong to this class of devices in the future [18, 40]. Therefore, the needed SQL DBMS implementation must fit the requirements of special-purpose embedded solutions. A viable DBMS implementation could be represented from the very portable and mature SQLite, which is released in public domain and is apt and open to innovation and research. Nevertheless, a major drawback in the SQL approach is the SQL database language itself. Notably, SQL falls short when it has to cooperate with AI. Although the SQL database language is declarative and a fourth-generation

programming language, it does not supply the full expressiveness needed, for example, for a complete first-order logic or modal logic inference problems. To cope with that, the real model that should be adopted and implemented soon is the RM in the original Codd's sense [10] as reviewed by some followers [11]. The RM, in general, can act as a higher-order logic representation language, so fit to cope with all problems and inherent limitations of SQL; for example, SQL has not a standard query for asking a list of the database tables or treat a table itself as a variable. Producers usually overcome this limit with proprietary extensions. Unfortunately, no lightweight implementation of the RM is still available. Until the availability of research efforts and results in the RM implementation, we are left with SQL and its shortcomings. Besides, SQL-based automation still can represent a great step forward in the paradigm for industrial process control right now.

Means and techniques to adapt SQL to AI have been already developed [1, 6, 38]. The use of SQL as a base technology for AI and automation is promising and is the straightest path toward managing the complexity of CPS challenges.

3 Factory and Process Automation Problem

A DBMS-centric approach for complex process automation was already looked into through the federation of MySQL and PostgreSQL technologies in 2008 in the developments of a European FP7 research project [8]. MySQL was used for embedded units as a moderately lightweight DBMS solution for embedded RTU boards, whereas PostgreSQL was used as the remote global data center. The major concept put forth there was the unifying role of the DBMS to host heterogeneous technologies for acquisition, actuation, and data processing. With suitable and simple adapters, any sensor and actuator, along with special-purpose computing units and machines, was connected in a whole unified DBMS informational center. Figure 1 shows the unifying

Figure 1: Unifying Role of the Database-Centric Approach.

role of DBMS-centric technology in an architecture that copes with the problem of the heterogeneity of the automation components.

Figure 1 expresses the main concept put forth here by showing how the several layers from enterprise systems, down to the shop-floor, are put into contact by the use of database queries. Each of the entities is provided with a DBMS infrastructure and can emit or receive database queries to communicate actions or information. Whereas for higher layers DBMS is a typical provision, here also the shop-floor is armed with such a capability. In this way, most of the interoperability problems typically encountered in the industrial playground can be overcome by apt choices on the semantics over exchanged and transmitted data through relational attributes specifications.

Note that the DBMS must be a distributed system, in particular a multidatabase system [34]. The distribution of the information and processing is carried on by lightweight synchronization mechanisms obtained by a well-calibrated data replication (explained in the following section). A distributed infrastructure is obtained when suitable mechanisms are used to propagate the updates on database tables across the distributed DBMS units. Therefore, the UPDATE or INSERT on a table due to a new acquisition is propagated to a networked DBMS that hosts a logic that needs the information update as input.

4 Distributed DBMS-Centric Methods

The database can be used as the unifying layer of abstraction. Currently, the similar solutions available on the market rely on more or less open object-oriented solutions (e.g. Java and Open Service Gateway Initiative) that can typically only scale down to $10 \div 50$ MB of program code on devices. The new Internet of Everything requires scalability at least below 1 MB. Notably, the SQLite database needs only 150 kb or less [35] when it is fully optimized. Scalability is obtainable with specific software engineering and embedded tool-chain technologies. Due to the technology currently available on DBMS that uses SQL implementations, the enforcement of the infrastructure here proposed requires at least the following prerequisites:

- (i) DBMS featuring transaction processing and indexing (although very limited) to achieve a high degree of reliability, performance, and capacity and
- (ii) Stored procedures, functions, and triggers that can perform external calls into the operating system facilities.

Although SQL standard manifests many inherent limitations, hints for database design that have arisen from experience on the field rely on a few golden rules, in particular for applications such as SCADA systems. A common trade-off in the design of an SQL database is the choice between a layout of tables oriented to easy and rapid data retrieval or inspection and tables oriented toward computational efficiency at the cost of a more complex inspection for synoptic data views (e.g. in human-machine interfaces or reports).

It can be noted that any new acquisition from a sensor is initially stored. However, as a major side effect, a trigger on an UPDATE or INSERT action for a *relvar* (relational variable) can pass on the last acquired value to an external procedure by means of interprocess communication (IPC). In particular, signals (in the Linux sense) can be effectively used in many cases. In case of Web technology, where a human interface is displaying and monitoring values, an HTTP POST command can be sent to the local or remote processes formerly subscribed to react to the interested *relvar* changes. In this way, a Web human-machine interface can provide the timely display of the last value of the *relvar* or, with the same principle, a controller can react by producing changes on other *relvars* that in turn may call for a physical action procedure. This allows every device and controller to be in contact through the data and their value records simply by querying the database.

4.1 Lightweight Database Synchronization Through Distributed Replication

Through the replication mechanism, a selective and optimized synchronization among the computing actors is obtained. Through appropriate configuration programming (in database language, of course), the set of

subscriber components asynchronously receive updates through the networking infrastructure. Information is propagated and communicated across the whole infrastructure through appropriate triggers, registered in the database tables. This allows every device and controller to be in contact through the data simply by querying the database. Each embedded database can synchronize centrally or with a peer neighbor by means of a lightweight replication mechanism. In SQLite, for example, these kinds of triggers are easily implemented. A trivial example of trigger code is provided, with the assumption that table `v1` has value and timestamp columns (attributes):

```
CREATE TRIGGER varpublish AFTER INSERT ON v1
BEGIN
SELECT publish_value(NEW.value, NEW.timestamp);
END;
```

The `publish_value()` procedure can be written in C, or in any other language, and might transmit the value to a distant component's database that requires it for its own device control purposes. Some assumptions can be made that allow synchronization efficiency. Furthermore, the creation of such a call to a procedure external to the database language can be completely avoided and substituted with a mere INSERT query, if the database has internal provision for addressing tables on a networked and distributed database file system. It can be assumed that a variable `xxx` is unique and has an absolute identifier (at present, the IPv6 addressing scheme appears to be the best identifier choice). The variable `xxx` might be associated to a physical object in the universe. The `xxx` identifier must anyway be unique and reserved forever (like a Web URI). It has to be assumed that the process that affects the values of variable `xxx` is unique, whether it is a mere sensor acquisition task or complex processing. It can further be assumed that `xxx` is a primary variable, as it is the only source of the information related to `xxx` in the whole universe, and so it is called a *Master* variable. A *Master* variable is a variable that is physically acquired by a device (e.g. embedded board, general-purpose computer, and software) or is generated through calculation from one or more acquired variables. *Slave* variables are then defined as those variables that need to be synchronized to a certain *Master* through replication. Of course, a process that affects a *Master* variable can also concurrently be fed as a *Slave* from other *Master* or *Slave* variables. A topological structure is then obtained that allows all of the sources of information to feed all of the distributed components that subscribe to the *Master* variables.

Note that the replication configurations are themselves database variables, as everything should be a relational variable in the RM. They can be shared and moved across the network as the processes that use them are also distributed but unique (similar to a Domain Name System).

4.2 Publish/Subscribe Paradigm and the IoT

The formerly explained replication technique addresses the major requirement for communication in CPS. In this context, the communication is the IoT. It is well known that a challenge posed by the IoT is the search for a unified and lightweight means of communications that allows machine-to-machine intelligent connection [2]. Breaking new ground in pervasive and distributed computing is the *swarmlet* concept. Swarmlets are applications and services that leverage networked sensors and actuators with cloud services and mobile devices. Their architecture is conceived to embrace heterogeneity instead of attempting yet another unreasonable standardization [25]. Also, the proposed DBMS replication technique is candidate to be easily compliant with the major architectures in IoT. The architecture proposal here follows a *publish/subscribe* scheme, where each RTU or software acts like a human being that only subscribes to interesting events/information and publishes relevant events or information. Prominent examples of publish/subscribe schemes are MQTT [29], robot operating system (ROS: <http://www.ros.org/>). A promising framework is the DDS (Data Distribution Service, at <http://portals.omg.org/dds/>). DDS addresses data in a manner similar to relational databases and recently has been considered its optimization for the IoT [5]. Furthermore, the Allseen Alliance puts forth the AllJoyn® Framework (<https://allseenalliance.org/>). The concept in common to all

the publish/subscribe schemes mentioned is the notion of the *topic*. Each node subscribes and some other publishes information on the *topic*. In our DBMS-centric proposal, the *topic* is simply the database content itself without intermediate language adapters: the unique language is the database language and its queries. All the information in the process is published to actors and any controller and optimizer can be built over the DBMS infrastructure and its semantic abstraction as a plug-in. Albeit in a different context, in Ref. [14], there is a quite detailed explanation of the advantages of such an active database-centric approach. In the distributed system, the DBMS must play the role of IPC. In Ref. [14], the IPC is mentioned but not used in a distributed sense. Indeed, in the example there developed, only one central DBMS computing unit was used. They used the term *resource adapter* to define a computing processing unit that is connected with physical actuators and sensors and that communicates with a centralized DBMS through the IPC extensions available on PostgreSQL – NOTIFY, LISTEN, and UNLISTEN commands. In the industrial distributed control system (DCS) context, a similar resource adapter role is performed by the RTUs. IPC through DBMS connects the RTUs and higher-level central units of the DCS system.

4.3 RTUs and Plug-Ins

Typically, an RTU in industrial control systems is a *holonic* unit that features some local procedural capabilities along with concurrent communication tasks. It is responsible both for real-time autonomous controlling actions and decisions and for communications with higher intelligence layers up to the enterprise management level. Holonic systems are relevant in enabling the multiagent system technology for intelligent distributed control of industrial plants [27]. For our purposes, an RTU device features an embedded DBMS and suitable replication mechanism (as previously explained). We need to identify and separate four parts in such an RTU structure: the input, the database tables, the logical and procedural part, and the output. In Figure 2, we identify the four holonic RTU parts. The input part contains the hardware and software that converts input information to an INSERT/UPDATE database query. DBMS tables and related

Figure 2: Significant Functional Parts of the Holonic RTU.

triggers constitute the second part. The database triggers launch the logic and algorithms relevant to the former event (with a local publish/subscribe scheme as well) on the third part. That might produce a new INSERT/UPDATE query as a result or a physical output to be handed off to the output part. Concurrently, information is published or received with the replication mechanism across the whole DCS. When a sensor acquires new data, it is put in the DBMS through an INSERT command. The new record is transmitted to other *holons* or to a higher-level (or central) DBMS by simply transmitting the same database query through the network. If some distributed business logic depends on that information, a database trigger calls the associated procedure. The output can in turn be propagated to other units. The same mechanism, in reverse order, is used for actuators.

In this computing infrastructure, the database tables are the globally shared storage part with their triggers for IPC. The components see the database tables as the unified and unique information interface available. With such an infrastructure, there is fertile land for intelligence and logic to grow and develop.

With this infrastructure available, algorithms can easily be plugged into the physical world as a back-end while they see a unified database front-end that enables controllers and other software to be seamlessly interconnected. Every algorithm developed is deployed as a *plug-in* of the system. Every plug-in can be immediately put into communication with the others. This enables a vast class of control topologies and hierarchies and any controller software is a plug-in. With the plug-in concept, an abstraction layer has been created that enables scientists and engineers to operate the control framework without being concerned about the low-level communication details. A plug-in can be uploaded, enrolled, and started by a Web-based graphical interface.

Figure 3 illustrates the workflow of plug-in development and deployment. What scientists or practitioners need are the knowledge and the semantics of the relvars (database variables) along with the controller source code. The controller source code can come from a simulated and calibrated Simulink® or Stateflow® model or any Matlab® or high-level programming language. Having in mind the input and output variables, the designer can edit the Integration File with the assignment of the inputs/outputs of the control algorithm to the corresponding database variables – the Integration File as a set of relations must be encoded as a database table as well. The plug-in file bundle (typically a compressed archive, as a .zip file) is uploaded by a Web page defined by a server that compiles and deploys the plug-in into the desired embedded boards or the cloud infrastructure. Plug-in runtime routines can definitely be started and stopped by a dedicated Web page. Note that, in principle, only a plain-text editor is a prerequisite to implement a plug-in (this is a reminder in some ways of the early days of the Web, and its universal accessibility by design through declarative language).

Figure 3: Plug-In Design and Deployment Workflow.

The syntax of the Integration File is itself can be very simple and intuitive. There are three main text sections: [REL], the relationships between input/output variables and the relvars in the database; [MAIN], the name of the main control function in the uploaded file bundle (to which the relationships in the [REL] part make reference); and [DEPS], the list of ancillary files that is called by the main function. The path of the files should be in the relative path form. This is an intuitive way to implement a controller from a set of source files, as it is possible to cascade or connect many plug-ins (with many topology choices) by mere reference to database variables.

A key technology and tool in the deployment workflow is the use of an appropriate tool-chain for the embedded coding of high-level scientific or declarative software. The new cloud-based approaches to tool-chain design and maintenance are of particular interest here, as indicated in Ref. [19]. In Figure 3, the source code of the program is then transformed into the procedures that are triggered by the DBMS to perform the computations (more details on this in the next section).

The plug-in architecture can implement an evolutionarily DCS intrinsically by the ability to generate descendant plug-ins with dynamically computed parameters or configurations. Each plug-in in the form of an agent (or more generally as an *actor* in the actor model [41]) can produce metaprogrammed code (i.e. code that produces other programs) that can be compiled and deployed in the system. Database records fully trace the system evolution, as every plug-in leaves its footprint on the data model. A plug-in can be generated and installed by a program that produces its code (and the *Integration File*) as an output. The code produced will depend on the events and conditions linked to the evolution of the controlled system. This represents an interesting new frontier for future studies that involve dynamic programming and evolutionary computation and their impact in AI and cognition. This can thus add new functionalities, choices, and versions of the plug-ins themselves, both dynamically and at run time. This possibility can represent a groundbreaking and major novelty for the control of industrial systems. Potentially new and more sophisticated agents can follow a *genetic* improvement as unmanned offspring of a previous generation of agents or controllers.

5 Expected Experimental Framework Results

5.1 Virtual Framework Development

The design, cosimulation, synthesis, and optimization of a production process is a complex task. Commercial software is already available and well known in existing systems based on manufacturing execution systems (MES). Recent research proposes open knowledge-driven MES [21]. State-of-the-art prototypes encompass multiagent systems (MAS) [4] and then process knowledge formalization and automation through a set of UML-based techniques [17] based on OMG standards and object-oriented approaches [33]. In the author's view, the prior selection of the programming language and the possible metamodels of its immaterial with respect to a framework that uses RM as all of them can be represented within the RM's expressiveness. With the database abstraction (in RM sense) and the implementation of the plug-in architecture presented in the previous section, all the models and metamodels can be virtually encompassed. This virtual framework is then made up of components connected to a DBMS or knowledge base and programmed through declarative database languages. By starting from a metamodel, a sequence of declarative queries and instructions can produce and interconnect a suitable and dynamic set of procedures.

Suitable APIs to database languages and DBMS already exist for the vast majority of procedural, functional, and logic programming languages. Current work is in progress from authors toward the formalization and implementation of a virtual framework for simulation and rapid prototyping and test of automation components, whether they are mere procedural or intelligent components and agents, through their interconnection in the plug-in infrastructure. With Matlab Simulink® and Stateflow® being quite de facto standard for researchers in the control community, the first realizations of the virtual framework will focus on these tools

for a first validations and assessment of the proposed methodology. These kinds of tools will at least allow the following three tasks to be performed easily:

- (i) Creating models of the production lines, environment, and energy-consuming components.
- (ii) Gathering the real data and calibrating the models through real incoming data.
- (iii) Using the output of the calibrated models to assess the KPI effectiveness and optimize through data mining on the raw not aggregated data. Optimized model and related control are deployed as a plug-in on the real plant through metacompilation on the embedded systems in the shop-floor and beyond.

The actual novelty here put forth is on paying due attention to the capabilities of embedded DBMS for the dynamical modeling and optimal operation of hybrid and complex systems such as continuous, batch, or discrete production processes [35].

A good starting point for the next experimental work and the realization of the virtual framework is the set of already available blocksets from the Simulink® environment that support the publish/subscribe scheme. Such components have been already developed, for example, in ROS (Matlab Robotics System Toolbox®), DDS (MathWorks provides Simulink® blocks and MATLAB® classes for RTI Connex DDS), and the RT-LAB Orchestra® product (<http://www.opal-rt.com>).

The common characteristic in the architecture of such kind of publish/subscribe software in Simulink® starts with the connection of subscriber *blocksets* inputs to one or more publisher blockset outputs. Subscribers and publishers role can be installed into the same node or agent (i.e. an independent computing process or device) to let them behave like peers in the communication. This brings about the opportunity for hierarchical or flat network topologies as well.

In this context, as mentioned in the previous section, a common notion is the *topic*, a communication channel that is available for subscription. In the case of distributed database architecture, the contents and information patterns in the topic have no restrictions in complexity and expressiveness as the associated communication queue may dynamically depend on the relation queried. In this case, the topic could be the whole database or a subset of authorized relations as well.

The publish/subscribe scheme can be achieved by forging a tailored new signal/bus object in Simulink® that transforms the blockset or plug-in outputs into database INSERT/UPDATE queries. On query execution, an IPC event triggers a SELECT query for the retrieval of the blockset input values. This way, the simulator tool is controlled by the DBMS. Note that this also allows hybrid virtual configurations such as the hardware-in-the-loop (HIL) techniques (see, for example, the VxWorks™ Async Interrupt and Async Task blocksets within the Real Time Workshop Toolbox®). Figure 4 shows the functional scheme of the signal object component under development.

5.2 Rapid Process Control Prototyping Through Simulation and Modeling

In Ref. [7], the authors showed how the plug-in concept promotes openness in the rapid prototyping of scientific software and how it breaks the entry barriers for small and medium enterprises in industrial appli-

Figure 4: Expected Functionality of the New Simulink Signal/Bus Element (Central Box).

cations. Through database abstraction, we can model the industrial process automation in two classes of components. The first class is constituted by the database relations (e.g. tables) and communication means (e.g. IPC and the signaling by triggers). The first class pertains to the methods already depicted in Figure 4. The second class is represented by the procedures or the intelligent transactions that implement the controlling algorithms relying on data inputs and outputs of formerly defined plug-ins. Indeed, a major distinction is made between the logic and the communication. Within the second class, the logic resides the design of all the structures that constitute a complete model of the physical processes involved in the production. In turn, the logic class is to be divided in two subclasses, one for the control and one for the models suitable for simulations and model-based control (e.g. in model predictive control techniques).

Figure 5 tries to render how the DBMS-centric infrastructure and related plug-ins are a useful scheme for the creation of tools for the rapid prototyping and simulation of a production plant. Figure 5 expresses the concept of the database as the fundamental and invariant ground entity enabling the architectural link between the physical and the virtual industrial plants (the lower and higher parts of Figure 5, respectively). DBMS is the common ground that is to be developed once and remains invariantly valid with passing from the physical realm to the virtualization framework. The DBMS is the fundamental and invariant ground enabling the architectural link between the physical and the virtual industrial plants (namely the digital twin of the plant). The dotted connection lines in the modeling framework (the top part in Figure 5) are not physical connections but associations with the suitable variable tables (i.p. relations in RM sense). These interconnection entities are eventually transformed back into physical communications through the deployment process of the plug-ins.

By switching between simulation and physical acquisition and control, we can seamlessly create models of the plant or calibrate the controllers as well. Furthermore, by abstracting the (business) logic blocks and the physical plant blocks over the ubiquitous database infrastructure, we can immaterially replace them with

Figure 5: DBMS Abstraction and Development Framework.

the plug-ins that model the control and business logic and the physical plant models in a virtual environment. These components can be simulated, tested, and calibrated in the suitable development framework and then deployed back into the corresponding physical RTUs.

The implementation of the logic blocks and algorithms through stored procedures or database triggers can be done in principle. However, in practice, it can be challenging, if of any help, with respect to using more appropriate procedural host languages (such as C or C++). A best practice suggested here is to separate clearly the data and knowledge infrastructure from the business logic and to let freedom of choice for expert software developers on the more appropriate development tool.

Once the sets of data input and output are determined, through the DBMS relations (tables), the connections and communications across the different business logic blocks can be abstracted and simulated. The algorithms constitute logic and procedures that produce outputs from the inputs as in the black-box approach (or states in other approaches). In the glue or wrapper code of the plug-ins, there reside the provisions for their interconnection and so the first class of components previously defined.

5.3 Implementing the System Network and Communication

This section presents some hints for the implementation of the key infrastructure that belong to the class of communication components. The fundamental structures are the relations and the trigger procedures that implement the IPC. Communications can happen on the same device or distributed over a network. A convenient event-based architecture is proposed along.

In Figure 6, it is shown a sketch of the communication network architecture. The main concept is that, in the background of this system-of-systems, we can usually identify many of the blocks that compose the machines, having them the role of an agent or RTU or field device component in the overall process control. Figure 6 contains a machine that has the capability to acquire data from the physical world by an acquisition unit (*Acquisition* block). Another machine can produce physical actuations (*Actuation* block), and others perform some internal processing or decisions through the combination of three computing (digital and discrete time) devices: *Dev1*, *Dev2*, and *Dev3*. The *Acquisition* block produces information and events for *Dev1* and *Dev2*. *Dev1* controls *Dev3* and *Dev2*, which in turn generate stimuli for the *Actuation*.

Figure 6: Device Network Over Distributed DBMS.

Note also the *Timer* device that typically is the time reference in a discrete digital system. It determines and drives the overall sampling and control time for the system. The *Timer* can be a physical clock device that is shared in the network among the devices that may need subscribing to it. In Figure 6, the blocks synchronous to the *Timer* are *Acquisition*, *Dev1*, and *Dev3*.

The arrows in Figure 6 are of two kinds: the solid ones correspond to the INSERT into the database (left part in Figure 4) and relate to the publisher action. The dashed arrows refer to the subscriber action (right part in Figure 4). The information that is exchanged is the last value of the N th database variable v_N . The information communication is started by the device fetching of a wake-up stimulus from the database through a signal (or message) generated by a database trigger as the interested variable value is updated. The signal is then handled and the receiver will start a SELECT request to obtain the current values of the input the corresponding database variables. In turn, after the processing task of a device is completed, the outputs are written into the database and trigger the subscribers bound to this event.

Note that the t variable is highlighted as a particular one. It is a variable that holds the time reference of the actions performed from the devices. Some devices need to be linked to this event and some do not. A distinction should be made between time-triggered devices (as filters, time counters, dynamic synchronous systems, etc.) and event-triggered devices where only the availability of new input is relevant as for *Dev2* (and, for example, in some asynchronous actuators).

In Figure 7, a flowchart for *Dev1* is shown. It exemplifies how the implementation of the communication is handled from the generic device. The *INPUT STAGE* and *OUTPUT STAGE* are the necessary key wrapper components that enable the realization of the virtualization concepts expressed in Figure 5.

The functionality of the wrapper components like in Figure 7 can be implemented easily in most of the programming languages, so the abstraction and portability are well guaranteed across most of the already existing development frameworks. Besides, the potential of the presented solution, relying mostly on database language, is best expressed for embedded tiny and low-cost devices. In Ref. [35], the authors described an experiment of this kind of device architecture for a device of a few dollars and low resources with an SQLite embedded database on-board.

Many different implementations of the components of Figure 7 are possible. The details of them depend on the DBMS used, on the database language, and on the host language connected to them and the hardware. In the example of Figure 7, we used the SQLite syntax. The queries there shown need the

Figure 7: Device Network Over Distributed DBMS.

definition of a database structure. The set of the minimal database tables (relations) and triggers that provide basic the functionality here required are provided compactly through SQLite query syntax in the following code:

```
CREATE TABLE "publishers" ("Device_Name" VARCHAR, "Var_Name"
    VARCHAR, FOREIGN KEY("Var_Name") REFERENCES vars("VarId"));
CREATE TABLE "subscribers" ("Device_Name" VARCHAR, "Var_Name"
    VARCHAR, FOREIGN KEY("Var_Name") REFERENCES vars("VarId"));
CREATE TABLE "t" ("Timestamp" INTEGER DEFAULT (0));
CREATE TABLE "v1" ("Value" VARCHAR DEFAULT (null),
    "Timestamp" INTEGER DEFAULT (null));
CREATE TABLE "v2" ("Value" VARCHAR DEFAULT (null),
    "Timestamp" INTEGER DEFAULT (null));
CREATE TABLE "v3" ("Value" VARCHAR DEFAULT (null),
    "Timestamp" INTEGER DEFAULT (null));
CREATE TABLE "v4" ("Value" VARCHAR DEFAULT (null),
    "Timestamp" INTEGER DEFAULT (null));
CREATE TABLE "v5" ("Value" VARCHAR DEFAULT (null),
    "Timestamp" INTEGER DEFAULT (null));
CREATE TABLE "v6" ("Value" VARCHAR DEFAULT (null),
    "Timestamp" INTEGER DEFAULT (null));
CREATE TABLE "v7" ("Value" VARCHAR DEFAULT (null),
    "Timestamp" INTEGER DEFAULT (null));
CREATE TABLE "vars" ("VarId" VARCHAR PRIMARY KEY NOT NULL,
    "Description" TEXT, "Master_Address" VARCHAR, "Type" VARCHAR);
```

Along with the previous SQLite code, describing the structure of the minimal database for the example of Figures 6 and 7, the contents of the two important publishers and subscribers tables are depicted in Figure 8. The tables in Figure 8 are equivalent to a circuit nodes list where there is specified in relational form the topology of the system-of-systems.

Publishers	
Device_Name	Var_Name
Timer	t
Acquisition	v1
Acquisition	v2
Dev 1	v4
Dev 1	v5
Dev 1	v3
Dev 2	v7
Dev 3	v6

Subscribers	
Device_Name	Var_Name
Acquisition	t
Dev1	v1
Dev1	v2
Dev1	t
Dev2	v2
Dev2	v3
Dev2	v4
Dev3	v5
Dev3	t
Actuation	v6
Actuation	v7

Figure 8: Minimal Contents of Publishers and Subscribers Tables for the Proposed Example.

5.4 Distributed Embedded Database Infrastructure and Real-Time Issues

In the previous section, the relation (table) “vars” contained the “Master_Address” attribute (or column). This is the provision for a possible network address and a communication protocol associated. This is a requirement for a network of the kind needed in the context of the IoT and CPS.

Currently, tiny DBMS as SQLite do not provide the capability to querying relations partitioned into networked slices of n -tuples subsets. Future work should provide this capability to let the querying process automatically search from distributed tables over a network. This feature requires an effort in case of real-time constraints due to both the network infrastructure (as in industrial EtherCAT® specifications) and the real-time temporal provisions from the database itself.

Present DBMS implementations suffer from “the query that dims the lights” problem: the access time and energy used to retrieve and store information is neither constant nor deterministic. RM implementation promises to overcome this gap, but few results are available to date for apparently commercial and technical reasons [12].

Database language-based programming techniques must allow inherent time- and event-based computation, which enable asynchronous dynamics that take place on different temporal and spatial scales. Real time, both in the sense of the timing validity and the consistency of the data, must guarantee a deterministic access time for transactions with timing constraints: data are to be considered valid only for specific time intervals by assigning them time semantics [13]. Moreover, the synchronous operations are to be guaranteed to render consistent a scheme like the proposed one in the example of Figure 6, where all the operations triggered have to be accomplished within the constraints of the transmission and propagation time of the queries on the network plus the processing time constraints put on the devices involved.

The RM-based techniques in Ref. [13] can possibly encompass and integrate the several temporal logic formalizations such as HRELTL (Hybrid Linear Temporal Logic with Regular Expressions) [9] or MTM (Metric Temporal Logic) as surveyed in Ref. [32] and the references therein. Moreover, the same techniques in Ref. [13] are suitable for event calculus in knowledge representation contexts [3]. The choice of the clock system and so the sampling period along with its event propagation across a network of dispersed components is still a challenge for the effective feasibility of networked event-based schemes [26] and it is matter for future research.

6 Role and Positioning of the Database-Centric Approach

This section provides an outlook on how the proposed technique and methodology can be put in relation to existing promising technologies and classifications for distributed CPS automation that integrate IoT. Discussion will involve the grade of centralization, of the compliance with respect to the correct definition of smart objects [16], and some impingements with respect to relevant security issues in IoT [15].

6.1 Comparison to State-of-the-Art Intelligent Automation Frameworks

The methodology proposal here draft is an enabling technology for the harmonization of state-of-the-art in IoT and industrial automation under the CPS vision. It is not intended as a replacement of other technologies but a tool and aid that can be introduced incrementally into relevant parts of existing technologies. In Table 1, a summary is provided for comparison of our approach to two state-of-the-art approaches in industrial intelligent automation and IoT. A full discussion on the integration of existing architectures can be found in Ref. [37]. Here, we focus on two, in the authors’ view, promising research frameworks that gather inside most of the features desired in the field: the ADACOR MAS [4] and the swarmlets architecture [25]. Whereas ADACOR

Table 1: Comparison of the Database-Centric Approach and State-of-the-Art Industrial and IoT Solutions.

Aspects of technology		Database-centric approach		ADACOR [4]	Swarmllets [25]
Cyber: logic, intelligence, knowledge, and reasoning	Pros	Handled with RM language capable of modal logic and declarative expressiveness		Full FIPA capabilities with mature JADE framework. Interoperates with other frameworks	Relying on existing frameworks through interoperability
	Cons	Lightweight implementation of RM not available		Java, XML, and FIPA not easily deployed on tiny devices. Database needed at back-end	No explicit and specific solutions for intelligence. Database needed at back-end
Physical: contact with physical world and devices	Pros	Triggers and procedures for connection to device-level middleware		Mature XML and object-oriented programming APIs for low-level interoperability (e.g. OPCUA)	Optimized APIs and great interoperability of the accessors
	Cons	RM procedures not mature, fallback on SQL extensions		XML and OO overheads cannot scale on tiny devices	Accessors configuration tends to be centralized
System: communications, internetworking	Pros	Simple expressive text over any kind of transport layer		FIPA standard, very flexible and expressive	JSON more lightweight than XML over any transport protocol
	Cons	Security and QoS issues. Byte rate not optimized		FIPA not yet completely optimized in this sense	Restricted/rigid semantics of messages
Engineering strength: update, adaptation, recovery, upgrade	Pros	Community to focus only on database improvement, giving strong engineering, versioning, and dependability. SQLite mature and widespread		High maturity of technology, great flexibility. Good support and diffusion of the involved technologies across developers	High standardization through Web reference technologies. The engineering of components are based on the Web browser concept
	Cons	Too many SQL and too few RM implementations. No reference solution		Complex middleware frameworks to maintain with numerous components	Same weaknesses as of Web browser technologies but none in particular
Cursor of centralization: (1) centralized, (2) hierarchical, (3) modified hierarchical/hybrid, and (4) heterarchical /P2P, M2M		Can cover any kind of configuration relying on interoperable semantic domains created on the database schema. Focus on range from 2 to 4		The solution is typically suited for decentralized architectures ranging from 2 to 4 dynamically	Mainly useful for heterarchical IoT configurations. Its focus is on four
Alternate White and Gray Cells are for Pros and Cons, Respectively.					

is well known and documented in industrial field encompassing holonic, bionic, and fractal integrated management systems at once, swarmlets are quite new, but the research there involved seems to convey the experience coming from the automotive in IoT and CPS sense (e.g. Ptolemy II and AUTOSAR). The database-centric approach is proposed comparatively to help harmonizing the technologies and so the results coming from the two fields. A summary of pros and cons (white and gray alternating table cells) discussion of the major features in the solutions has been provided as a reference for future work in Table 1. The first to third rows compare the technological aspects pertaining to the cyber, physical, and system contexts, respectively. The last row discusses the grade of centralization with a cursor from 1 to 4.

6.2 New Role of Databases for Smart Objects

In Ref. [16], a useful review of the definition of smart objects in the context of IoT can be found. A major aspect covered there is a classification of the several kinds of objects encompassed by IoT with respect to their capability to feature intelligence. Following Meyer's classification (see Ref. [16] and the references therein), three dimensions of intelligence have been defined: *level*, *location*, and *aggregation*. The database-centric architecture has the potential to cope with all those different dimensions of the intelligence. In particular, as far as an object features an active database on-board, it can be classified as smart – of course, this is a sufficient but not necessary condition. In Table 2, some hints are provided to describe how database-centric architecture copes with the smart objects intelligence classification in Ref. [16].

6.3 IoT Security from a Database-Centric Perspective

In Ref. [15], the rising security issues provoked by the growth of interconnected devices for the IoT is analyzed. To worsen the problems, the numerous devices connected, for the greatest part, are destined to resource limits and constraints, also in the future [18, 40]. This means that the on-board capabilities to handle the several aspects of security are severely limited. In particular, it happens with RFID devices, which are usually the smallest objects in the IoT [15]. In considering all the layers of the IoT that scale from the lowest sizeable devices up to remote data centers, currently security is addressed both by hardware and software solutions, ranging from secure identification key generation and its use for authentication and cryptography to the classification of information. Security methods can be also divided into two major categories of policies: active and passive. The active policy intervenes to avoid any intrusion or harm before it happens. Usually, to enforce this policy is costly and requires lots of resources and energy. It can be metaphorically associated to a safe. On the contrary, passive policy lets the malicious fact happen but guarantees that there are enough information and traces to catch and condemn the offender. It acts as a deterrent, as a security cam.

From the database-centric point of view, security is not to be considered a specific specialty. Databases cannot add more security, apart from the contexts in which they are already used as infrastructure. Nevertheless, the use of embedded database can scale down and distribute more some of the security capabilities usually reserved to higher and centralized levels. Database is more prone to a passive policy, for its natural data-storing capabilities. Besides, active distributed databases have a great role in the classification of information and authorization of some of the data actively enforced by constraints in the schemas [34].

Indeed, an important requirement of a distributed DBMS is the ability to support semantic data control, i.e. data and access control using high-level semantics. Semantic data control typically includes view management, security control, and semantic integrity control [34] that ensure that authorized agents perform correct operations on the database. Views, security constraints, and semantic integrity constraints can be defined as ECA rules that actively impede the violation of some rules by an agent program and generally implies the rejection of the effects of that program. In Figure 9, we show pictorially the foreseen area of intervention of the database-centric approach with respect to the IoT levels of the devices and security. Of course,

Table 2: Definition of Object's Intelligence and Implementation.

Intelligence dimension	Levels	Definition freely adapted from [16]	Database-centric implementation
Level of intelligence: how much an object is intelligent and smart	Information handling	Any smart object manages information gathered and received from nonsmart objects possibly by classifying, treating, structuring, and enriching the raw data	Any object holds structured data with semantics added by the RM's schema. Data are received as queries, in textual form or transformed into queries when triggered by an update event
	Notification of the problem	At this level, objects do not feature free will. They have a procedural capability to notify events and alarms	By ECA rules, at the arrival of new data, a condition is checked against. If the condition is true, a trigger is fired to start a notification procedure
	Decision making	This is the highest level of intelligence, not merely procedural. The object has free will and can take decisions autonomously	Relying on the expressiveness of the RM, logic programming, descriptive logic, and modal logic can be implemented through database languages
	Intelligence through the network	All the reasoning is performed outside of the object from the networking with external controller agents	The object simply receives commands and data through replication (see Section 4.1) and communication. It does not reason over the data it receives
Location: where intelligence resides	Intelligence in the object	The object can perform some reasoning by itself without relying on external communication	With the data and the semantics added from the RM, reasoning on local schema and data is possible
	Combined intelligence	Objects rely both on autonomous reasoning and interaction with other agents in the network	Data and schemas can be updated and transmitted to change the knowledge base of a particular object interacting in a society of peers
Aggregation: how intelligence emerges from the components	Intelligence in the item	These objects have the information handling and notification capabilities. They rely on a fixed set of immutable components	This is the situation in which the schemas, the structure of the communications, and ECA rules do not change during object's lifetime
	Intelligence in the container	Objects act as a proxy and broker with the capabilities of information handling, notification, and decision with respect to a dynamic set and status of their owned components	The connections and the set of ECA rules are modified dynamically to reflect the status and the quality of the components. The object can assume the role of coordinator
	Distributed intelligence	This is the category that fuses the two previous. Object can be both container or item with respect to a dynamically assigned role among peers depending on the current goal and task	By merging the two former capabilities, the database-centric approach encompasses completely the actor model in Ref. [41] and the multidatabase systems integration of global and local schemas [34] by shaping the fragments of information on purpose in a dynamical way

Figure 9: Area of Intervention of the Database-Centric Approach for Security.

the new approach can bring some complementary improvements but does not cover the whole aspects and range of problems and devices in the IoT.

7 Conclusion

This paper proposes a viable enabling technology for the harnessing of some of the complex issues in the incoming scenario of industrial automation where the smart features of CPS are going to play a leading role. Based on the lessons learned, state-of-the-art in the recent literature, and commercial trends, a database-centric architecture is proposed as a suitable and simple technology solution for the problems in networking and control of factory and process automation of the future.

The proposed technique focuses on the development of simulation framework, modeling, and rapid prototyping tools by leveraging the best available standard approaches and technologies. The potential of this methodology is candidate to express itself increasingly more in the medium term. In fact, the current trend presents the development of highly distributed system-of-systems implemented into swarms of tiny embedded and disappearing computing devices.

This work presented the guiding lines of the approach with examples and hints for future research. It constitutes the basis of a currently undergoing realization of a virtual framework for some comparative and quantitative studies between the proposed approach and the state-of-the-art in industrial playground. The validation of the methodology is expected to address the total ownership costs of the technology, real-time issues, and performance.

A strength of the approach is the capability to incrementally wrap, adapt, integrate, and retrofit existing state-of-the-art solutions without affecting the major structure and architectures of already valid approaches. In the short term, a set of new Simulink® tools are going to be developed for the expected development framework along with tests of advanced control techniques for the KPI-based optimization of production and energy efficiency of real processes.

A major drawback in this presented work is the current unavailability of a database language with complete RM support and that was suitable for tiny implementation. The SQL language examples here proposed have still major drawbacks due to the limits in expressiveness that RM adoption should overcome easily. A future work suggested is the development of scientific research for the enforcement of the full RM with

efficient practical implementations. Nonetheless, the current available technology and the proposed architecture can still provide in parallel clear methodological results.

Bibliography

- [1] A. AlAmri, The relational database layout to store ontology knowledge base, in: *International Conference on Information Retrieval & Knowledge Management (CAMP)*, IEEE, pp. 74, 81, 2012.
- [2] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari and M. Ayyash, Internet of Things: a survey on enabling technologies, protocols and applications, *IEEE Commun. Surv. Tutorials* **17** (2015), 2347–2376.
- [3] A. Artikis, M. Sergot and G. Paliouras, Reactive reasoning with the event calculus, in: *Proceedings of the 21st International Workshop on Reactive Concepts in Knowledge Representation (ReactKnow 2014)*, Prague, Czech Republic, August 18–22, pp. 9–15, 2014.
- [4] J. Barbosa, P. Leitão, E. Adam and D. Trentesaux, Dynamic self-organization in holonic multi-agent manufacturing systems: the ADACOR evolution, *Comput. Ind.* **66** (2015), 99–111.
- [5] K. Beckman and O. Dedi, sDDS: a portable data distribution service implementation for WSN and IoT platforms, in: *IEEE WISES 2015, Proceedings of 12th International Workshop on Intelligent Solutions in Embedded Systems*, October 29–30, pp. 115–120, 2015.
- [6] P. Bhatia, N. Khurana and N. Sharma, Intuitive approach to use intelligent database for prediction, *Int. J. Comput. Appl.* **83** (2013), 36–40.
- [7] A. Bonci, S. Imbrescia, M. Pirani and P. Ratini, Rapid prototyping of open source ordinary differential equations solver in distributed embedded control application, in: *10th International Conference on Mechatronic and Embedded Systems and Applications IEEE/ASME*, pp. 1–6, 2014.
- [8] CAFE Project, EU FP7-KBBE, g.a. 212754, *Deliverable. D.8.6: The Integrated System and the Software Main Frame*, 2008.
- [9] A. Cimatti, M. Roveri and S. Tonetta, HRELTL: a temporal logic for hybrid systems, *Inf. Comput.* **245** (2015), 54–71.
- [10] E. F. Codd, *The Relational Model for Database Management: Version 2*, Addison-Wesley Longman Publishing Co., Boston, MA, USA, 1990.
- [11] H. Darwen, *An Introduction to Relational Database Theory*, Ventus, 2012, Available at: <http://bookboon.com>.
- [12] C. J. Date, *Go Faster! The TransRelational™ Approach to DBMS Implementation*, 2011, Available at: <http://bookboon.com>.
- [13] C. J. Date, H. Darwen and N. Lorentzos, *Time and Relational Theory: Temporal Databases in the Relational Model and SQL*, Morgan Kaufmann Series in Data Management Systems, Elsevier Inc., 2014.
- [14] W. O. De Moraes, J. Lundstrom and N. Wickstrom, Active in-database processing to support ambient assisted living system, *Sensors* **14** (2014), 14765–14785.
- [15] P. Gaona-García, C. Montenegro-Marin, J. D. Prieto and Y. V. Nieto, Analysis of security mechanisms based on clusters IoT environments, *Int. J. Interact. Multimedia Artif. Intell.* **4** (2017), 55–60.
- [16] C. G. García, D. Meana-Llorián and J. M. C. Lovelle, A review about smart objects, sensors, and actuators, *Int. J. Interact. Multimedia Artif. Intell.* **4** (2017), 7–10.
- [17] S. Guermazi, S. Dhouib, A. Cuccuru, C. Letavernier and S. Gérard, Integration of UML models in FMI-based co-simulation, in: *Proceedings of the Symposium on Theory of Modeling and Simulation, Society for Computer Simulation International*, 2016, p. 7.
- [18] O. Hahm, E. Baccelli, H. Petersen and N. Tsfites, Operating systems for low-end devices in the Internet of Things: a survey, *IEEE Internet of Things Journal* **3** (2016), 720–734.
- [19] J. Hausladen, B. Pohn and M. Horauer, A cloud-based integrated development environment for embedded systems, in: *Mechatronic and Embedded Systems and Applications (MESA), IEEE/ASME 10th International Conference on*, September 10–12, 2014, pp. 1–5.
- [20] W. He and L. D. Xu, Integration of distributed enterprise applications: a survey, *IEEE Trans. Ind. Informatics* **10** (2014), 35–42.
- [21] S. Iarovy, W. M. Mohammed, A. Lobov, B. R. Ferrer and J. L. M. Lastra, Cyber-physical systems for open-knowledge-driven manufacturing execution systems, *Proc. IEEE* **104** (2016), 1142–1154.
- [22] Inductive Automation, *SQL: The Next Big Thing in SCADA, How SQL is Redefining SCADA*, White Paper, 2012, Available at: <http://inductiveautomation.com>.
- [23] S. M. Jeon and K. Gitae, A survey of simulation modeling techniques in production planning and control (PPC), *Product. Plann. Control* **27** (2016), 360–377.
- [24] Z. Jia, B. Iannucci, M. Hennessy, S. Xiao, S. Kumar, D. Pfeffer, B. Aljedia, Y. Ren, M. Griss, S. Rosenberg, J. Cao and A. Rowe, Sensor data as a service – a federated platform for mobile data-centric Service Development and Sharing, *emphServices Computing (SCC)*, in: *IEEE Int. Conf. On*, pp. 446–453, 2013.
- [25] E. Latronico, E. A. Lee, M. Lohstroh, C. Shaver, A. Wasicek and M. Weber, A vision of swarmlets, *Internet Comput. IEEE* **19** (2015), 20–28.

- [26] E. A. Lee, The past, present and future of cyber-physical systems: a focus on models, *Sensors* **15** (2015), 4837–4869.
- [27] P. Leitao, V. Marik and P. Vrba, Past, present, and future of industrial agent applications, *Ind. Inform. IEEE Trans.* **9** (2013), 2360–2372.
- [28] P. Leitao, J. Barbosa, M.-E.C. Papadopoulou and I. S. Venieris, Standardization in cyber-physical systems: the ARUM case, in: *Industrial Tech. (ICIT), IEEE Int. Conf.*, pp. 2988–2993, 2015.
- [29] D. Locke, Mq telemetry transport v3. 1 protocol, specification, *IBM developerWorks Technical Library*, 2010.
- [30] W. Mahnke, S. Leitner and M. Damm, *OPC Unified Architecture*, 2009.
- [31] M. Nickel, K. Murphy, V. Tresp and E. Gabrilovich, [A review of relational machine learning for knowledge graphs](#), *Proc. IEEE* **104** (2016), 11–33.
- [32] P. Nuzzo, A. L. Sangiovanni-Vincentelli, D. Bresolin, L. Geretti and T. Villa, [A platform-based design methodology with contracts and related tools for the design of cyber-physical systems](#), *Proc. IEEE* **103** (2015), 2104–2132.
- [33] OMG, *Object Management Group Specifications*, 2016, Available at: <http://www.omg.org/spec>.
- [34] M. T. Özsu and P. Valduriez, *Principles of Distributed Database Systems*, Springer Science & Business Media, Springer, New York, Dordrecht, Heidelberg, London, 2014.
- [35] M. Pirani, A. Bonci and S. Longhi, A scalable production efficiency tool for the robotic cloud in the fractal factory, in: *Proceedings of the 42nd IEEE Industrial Electronics Conference (IEEE IECON2016), Florence, Italy, October 24–27, (2016)*, in press.
- [36] A. Rajhans, A. Bhave, I. Ruchkin, B. H. Krogh, D. Garlan, A. Platzer and B. Schmerl, Supporting heterogeneity in cyber-physical systems architectures, *IEEE Trans. Automat. Control* **59** (2014), 3178–3193.
- [37] N. Schmidt, A. Lüder, R. Rosendahl, D. Ryashentseva, M. Foehr and J. Vollmar, Surveying integration approaches for relevance in cyber physical production systems, in: *Emerging Technologies & Factory Automation (ETFA), 2015 IEEE 20th Conference on*, 2015, pp. 1–8.
- [38] H. Schuldt, Agent and databases: a symbiosis? in: *Cooperative Information Agents XII, 12th International Workshop*, September 10–12, 2008, CIA, Prague, 2008.
- [39] F. Stiel, T. Michel and F. Teuteberg, Enhancing manufacturing and transportation decision support systems with LCA add-ins, *J. Cleaner Product.* **110** (2016), 85–89.
- [40] S. M. Trenkwalder, Y. K. Lopes, A. Kolling, A. L. Christensen, R. Prodan and R. Groß, OpenSwarm: an event-driven embedded operating system for miniature robots, in: *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pp. 4483–4490, 2016.
- [41] C. A. Varela and G. Agha, *Programming Distributed Computing Systems: A Foundational Approach*, The MIT Press, Cambridge, MA, US, 2013.
- [42] VDI/VDE Society Measurement and Automatic Control, Status Report: Reference Architecture Model Industrie 4.0 (RAMI 4.0), 2015.
- [43] S. Yang, T. Khot, K. Kersting and S. Natarajan, Learning continuous-time Bayesian networks in relational domains: a non-parametric approach, *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, 2016.