Advanced Engineering Informatics 32 (2017) 113–125

Contents lists available at ScienceDirect

Advanced Engineering Informatics

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

Private local automation clouds built by CPS: Potential and challenges for distributed reasoning

Borja Ramis Ferrer *, Jose Luis Martinez Lastra

Tampere University of Technology, Fast-Lab., P.O. Box 600, FIN-33101 Tampere, Finland

ARTICLE INFO

Article history:
Received 19 March 2016
Received in revised form 23 January 2017
Accepted 25 January 2017
Available online 8 February 2017

Keywords:
Industrial automation
Cyber-physical systems
Local cloud
Ontology
Distributed reasoning

ABSTRACT

The employment of cyber-physical systems allows the control of processes in modern production lines. On the other hand, several research works have recently presented how ontology-based knowledge representation can be a suitable method for modelling industrial systems. However, system models are located far away from where the data is generated which adds complexity for cross-domain communications and resource management. Current embedded devices can encapsulate ontological models that can be accessed as local resources. This article presents the integration of interconnected devices as the computational nodes of a cloud which is private and local. In this way, functionalities, such as knowledge management and process control can be performed closer to the industrial equipment. Moreover, this research work discusses the potential and challenges for performing distributed reasoning in the private local automation cloud. In addition, the article describes main aspects of the system architecture and the behaviour of the networked embedded devices in the cloud. The research work results will be used as a high-level roadmap for further system implementation.

1. Introduction

The industrial automation has evolved passing through several generations of embedded devices, which are used for process control at the factory floor. This evolution has resulted in the development of modern Web Service (WS) enabled controllers, such as the S1000 from Inico Technologies,[1] which allows the implementation of the Service Oriented Architecture (SOA) paradigm [1] in modern production lines, as described in [2].

During last decades, industrial control systems have employed different standards for implementing multiple solutions. For example, the IEC-61131-3 [3] and the IEC-61499 [4] can be used for modelling and information exchange in distributed control systems. Actually, one of the main challenges in modern industrial control systems is the management of a large amount of data that is exchanged between system components. One solution is the employment of IEC-61499 for modelling automation systems within Function Blocks (FB) networks [5].

In addition to standards, the industrial domain has included the use of Knowledge Representation (KR) for implementing knowledge-based systems. These systems are beneficial in the industrial field for:

- overcoming the interoperability issues between system modules within data mappings [6,7]
- monitoring system status at runtime [8], and
- implementing knowledge-driven approaches, in which ontology-based KR enables the modelling and decision-making support in systems [9,10]

Recent research in the industrial automation domain present system architectures in which the management and location of the knowledge is performed away from the shop floor physical machines. As an example, the eScop project[2] (Embedded systems Service-based Control for Open manufacturing Project) proposes a possible architecture in [11]. The implementation of the project architecture and principles is described in [12]. Similar to the ISA-95 standard automation pyramid,[3] the eScop architecture presents that machines reside at the lowest level (i.e. device or shop floor level) of a hierarchical multi-layer structure. Then, the Industrial machinery is isolated from the process of information, which is performed by production management level systems; e.g., the

* Corresponding author.
E-mail addresses: borja.ramisferrer@tut.fi (B. Ramis Ferrer), jose.lastra@tut.fi (J.L. Martinez Lastra).

[1] http://www.inicotech.com/.

[2] http://www.escop-project.eu/.

[3] http://isa-95.com/.

http://dx.doi.org/10.1016/j.aei.2017.01.007
1474-0346/© 2017 Elsevier Ltd. All rights reserved.

Manufacturing Execution System (MES). Once the information is processed, the upper layers orchestrate the invocation of operations that are described in Remote Terminal Units (RTUs), which are embedded devices physically connected to the machinery. Therefore, RTUs facilitate the implementation of the Cyber-Physical Systems (CPS) concept, which is defined as the integration of computation (cyber systems) and controlled equipment (physical systems) [13]. Basically, RTUs act as a gateway for allowing cross-layer communication between higher level systems and the industrial equipment which is located at the device level. A representation of the pyramidal structure is illustrated in Fig. 1.

As depicted in the above diagram, industrial enterprise systems are distributed in different levels; i.e. factory shop floor, supervisory control, production management and business management. RTUs (and also PLCs) are the interface between the device level and control level. The flow of information can be (i) horizontal, i.e. exchange of information between systems residing on same level or vertical (ii) cross-layer communication between systems located at different levels.

This research is motivated by the fact that the computational resources of the embedded devices used in the industrial domain have increased. Therefore, now it is possible to add more functionality that currently reside in higher levels of the automation pyramid into the devices which are located at the factory shop floor level. Moreover, this research work is motivated for the reduction of the high cost of integration in the industrial automation field. The interconnection of autonomous, flexible and configurable devices should result in a decrement of integration efforts because systems that currently reside on higher levels of the automation pyramid will be replaced by functionalities hosted in the embedded devices located at the factory shop floor.

This research proposes a new architecture for the hierarchical structure implemented in industrial automation enterprises, in which embedded devices will be computational nodes that cooperate for making decisions, controlling industrial processes and other functionalities that are currently managed and coordinated away from the device level. In fact, the proposed distributed system network that is composed by embedded devices creates a private local automation cloud, which is capable of receiving and solving external requests about services and computation resources. On the other hand, the presented approach describes the potential and challenges of using embedded devices that integrate their local resources in order to accomplish distributed reasoning. Therefore, the contribution of this research is not only to propose of an alternative for the conventional architecture in industrial enterprises, but also to provide the design of the private automation cloud, ontological model and behaviour between devices for exchanging information.

The rest of the paper is structured as follows: Section 2 presents literature and industrial practices of aspects and areas of interest for the presented architecture to be developed and implemented in the field of factory automation. Section 3 introduces the principles of the private automation cloud and describes the encapsulated knowledge and behaviour of devices for solving incoming requests. Then, Section 4 discusses the potential and challenges of the presented approach. Finally, Section 5 presents the conclusion and further work of the research.

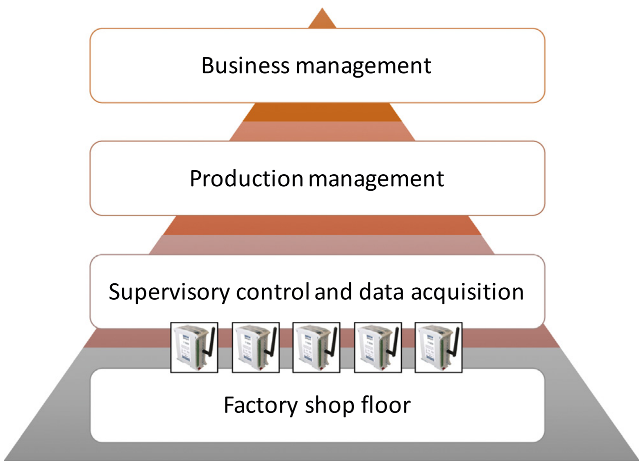


Fig. 1. A representation of the ISA-95 automation pyramid, including WS enabled RTUs.

2. Literature and industrial practices

2.1. Cloud computing definitions and some concepts

Cloud computing (CC) is a promising paradigm that is currently applied in several domains, such as industrial automation domain. In a nutshell, CC allows abstracting and storing computational resources, as well as providing on-demand access to those resources. CC is related to other research areas as high-performance, utility and grid computing. As an example [14], offers a detailed comparison between grid and CC. Fundamentally, CC can be defined as a computing model composed by networked elements, which control their own resources. The computing resources are hosted locally by each element, i.e. a server. Nevertheless, the cloud can be understood from the outside as a unique element that contains a large amount of cloud services that are used for responding to incoming service requests. In addition, cloud services are a type of WS [15].

There are three different service models (or categories) of CC: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS). Usually, IaaS, e.g. Google Compute Engine (GCE),[4] utilizes virtual computer resources that are outsourced by an organization [16]. IaaS users can remotely access and manage the data. Thus, IaaS is also known as HaaS (Hardware-as-a-Service). Secondly, PaaS is a service category that permits developers to implement their applications on a cloud. For example, Microsoft Azure[5] provides both IaaS and PaaS. Finally, the SaaS, e.g. Google Gmail,[6] service model allows users to access vendors' applications within the web. It should be noted that way of accessing to PaaS and SaaS is similar [17].

On the other hand, CC can be deployed mainly as a private, public or hybrid clouds, which are compared in [18]. A private cloud is a cloud that can be hosted internally or externally but operated only for a unique organization. Then, private cloud services are accessible solely for users and third-party applications, which belong to the organization that owns the cloud. On the other hand, cloud services of a public cloud are accessible by any user. This is possible because service access is provided through a public network. Finally, hybrid clouds are composed by more than one cloud. This fact does not exclude the composition of private and public clouds in the same hybrid cloud. For example, hybrid cloud is beneficial when an organization requires more computing resources that the ones offered by its private cloud. Then, the creation of a hybrid cloud, by incorporating e.g. a public cloud with accessible computing resources, is a possible approach for meeting the requirements. The deployment of hybrid clouds implies developing federated capabilities, which is needed for linking distributed resources.

[4] https://cloud.google.com/compute/.
[5] http://azure.microsoft.com/en-us/.
[6] https://mail.google.com/.

2.2. Integration of cyber and physical systems

The integration of cyber and physical systems is known as CPS [13]. Recent advances on CPS permit cross-domain integration and Machine-to-Machine (M2M) communication in the industrial automation field. Moreover, according to [19], the emergence of CC in the industry, and its combination with CPS, will make it possible to achieve some advantages as scalability and flexibility of resources, among other features in future manufacturing systems. This is possible due to the success of SOA implementation in factory automation. For example [2,20], demonstrated that functionalities of shop floor equipment can be exposed outside through WS. Then, by the deployment of a cloud with access to the shop floor services, integration of equipment and the cloud will permit accessing to physical resources through cloud services.

The relevance of CPS is discussed e.g., in [21]. Aforementioned research work introduces the next generation of manufacturing industry, referred as the Industry 4.0, which is to be built by CPS. Fundamentally, any physical "*thing*" related to industrial processes will be connected to the Internet, realizing the principles of the Internet of Things (IoT), so it will be also integrated to cyber-systems. A formal overview of the standards and patents for the IoT, presented as a key enabler for the Industry 4.0 can be found in [22]. Furthermore, an example of architecture for implementing CPS on Industry 4.0-based manufacturing systems is presented in [23].

2.3. Knowledge representation in factory automation

Knowledge Representation (KR) is a part of the artificial intelligence concerned on describing the world and allowing computer systems to make decisions by using such descriptions [24]. Moreover, a Knowledge Base (KB) can be defined as an engineering artefact used to keep the knowledge representation. There is a large amount of KR formalisms e.g., ontologies, production rules or frames. A large amount of factory automation developments has used ontology-based KR for both system and product knowledge modelling as e.g., the research works presented in [25–29]. T. Gruber defined ontology as *"an explicit specification of a conceptualization"* [30]. Then, ontologies contain formal description about certain domain [31].

Although there are many semantic languages for modelling manufacturing systems [32], the Ontology Web Language (OWL)[7] is preferred for such descriptions due to its high degree of expressivity. Truly, OWL is a vocabulary extension of the Resource Description Framework (RDF)[8]. Then, it is possible to use RDF-based query languages such as the SPARQL Protocol and RDF Query Language (SPARQL)[9] and the SPARQL Update[10] for retrieving and updating ontological model data, respectively. This is an important fact for integration of computer systems because whenever a system needs some data described in the model, it can access it by querying the model. For example, the research work described in [33] presents a set of queries used for retrieving data from a production line ontological model. As it can be seen in aforementioned research work, OWL-S is an OWL-based model for service description.

On the other hand, reasoning is defined in [24] as "the formal manipulation of the symbols representing a collection of believed propositions to produce representation of new ones". This process is performed by semantic reasoners or reasoning engines operating on OWL models. A comparison between existing reasoning engines can be found in [34]. This is beneficial for systems implementing ontology-based KR because reasoners can deduce dynamically data sets that are not defined as a fact when the model is created. For instance, data mappings and classification of system instances in the model are tasks easily performable by semantic reasoning engines [35]. It should be noted that for supporting reasoner inferences, the use of the Semantic Web Rule Language (SWRL) allows the definition of rules on top of OWL models [8]. Moreover, the utilization of reasoning engines allows checking the inconsistency of ontological models, resolving conflicts and reducing redundancy of data. In this scope, a recent research work presented an ontological framework that supports decision making in industrial scenarios within the use of semantic rules created for domain-specific ontologies [10].

Several research works in the factory automation domain present solutions that implement KR in different kinds of architectures. For instance, one of the previous cited research works [33], combines SOA and KR. In fact, the same research team describe how to dynamically retrieve OWL-S descriptions of services in order to find the ones required to perform industrial process actions [36]. Besides its integration with SOA, KR is also implemented in different kind of architectures, such as event driven architecture as recently presented in [37].

2.4. Distributed reasoning

In distributed systems, networked computer systems coordinate their actions within the exchange of messages. In fact, the coordination of actions is done for solving incoming requests to the system. It should be noted that, in this case, resources are understood as the KB data of a computer system. Then, distributed reasoning is defined as a process performed by several computer systems, which exchange messages for integrating their resources.

The principal objective of distributed reasoning is to reason integrated KB data of different computer systems. Fundamentally, taking into account that each computer system owns a different local KB, the integration of several computer systems data is required for responding to requests that cannot be responded using only the resources of a single computer system.

Therefore, it can be stated that the implementation of this behaviour is a requirement for future private and local automation clouds allowing embedded devices to cooperate for solving problems jointly. An employable mechanism for such need has been already described in [38]. Aforementioned work presents an approach for managing distributed knowledge, which is encapsulated in embedded devices, aiming the integration of resources for solving incoming requests.

3. A private automation cloud built by CPS

This research work proposes the creation of a private automation cloud composed by embedded devices, which encapsulate knowledge and functionalities that nowadays are nested and managed above the device level. This cloud is private because is only accessible by allowed elements of a unique organization and resides locally in the same organization. The proposed approach is motivated by the evolution of devices currently available and used at shop floors for process control.

Nowadays, embedded devices have much more computational resources than previously. Hence, the performance of additional tasks beyond direct process control is now feasible. Some of these tasks, as the encapsulation of system knowledge, are directly downgraded from, in this case, production management to shop floor level. Nevertheless, the inclusion of new duties also implies the development of non-existing ones, as e.g., the implementation of algorithms that devices must be capable to execute for managing

7 https://www.w3.org/TR/owl-ref/.
8 https://www.w3.org/RDF/.
9 https://www.w3.org/TR/rdf-sparql-query/.
10 https://www.w3.org/TR/sparql11-update/.

knowledge encapsulation and knowledge integration, which is needed for distributed reasoning [38].

3.1. Private cloud architecture

The private automation cloud includes interconnected embedded devices which control production line processes. This means that the private cloud is directly connected to the physical equipment, integrating physical resources with cyber systems. A basic architecture representation of the system built by CPS is shown in following Fig. 2.

As it is explained in [2], the SOA paradigm can be implemented in actual production lines by the use of certain RTUs, e.g. S1000 from Inico Technologies shown in Fig. 2. This is possible using the Device Profile for Web Services (DPWS) stack that allows implementing WS in resource-constrained devices. Through this technology, embedded devices can be discovered and it is also possible to invoke described services [39], among other functionalities.

Then, the architecture shown in Fig. 2 represents how the shop floor equipment is connected to embedded devices, which control the machine processes. It should be noted that the connection machine-device depends on the machine that is controlled e.g., a conveyor can be connected within digital I/Os and robots via RS232. Aiming the illustration of how this is implemented in a real scenario, Fig. 3 shows RTUs employed by the FASTory line, which is an assembly line located at Tampere University of Technology facilities, for controlling processes. Fig. 3 shows two different views of FASTory line cells, which are capable to draw 729 different variants of mobile phones (FASTory line detailed description is included in [12]).

The objective of showing two different perspectives of FASTory cells is to show i) the major components of each cell (i.e. conveyor and SCARA robot) and ii) the connected RTUs. As it can be seen at the bottom right side of Fig. 3, the three S1000 controllers (that are identifiable by their blue led lights) are connected to the conveyor segment and robot of the corresponding cell. Moreover, one extra RTU is used as an energy meter to manage and monitor the energy

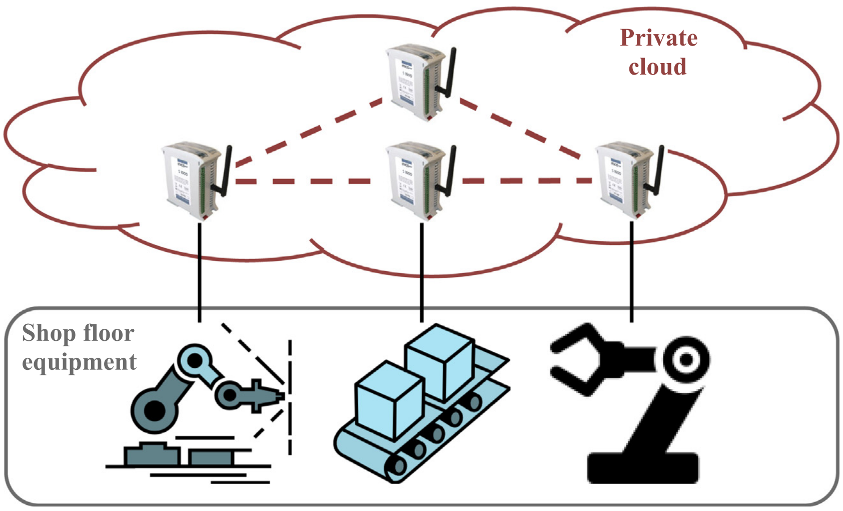


Fig. 2. The private automation cloud basic architecture.

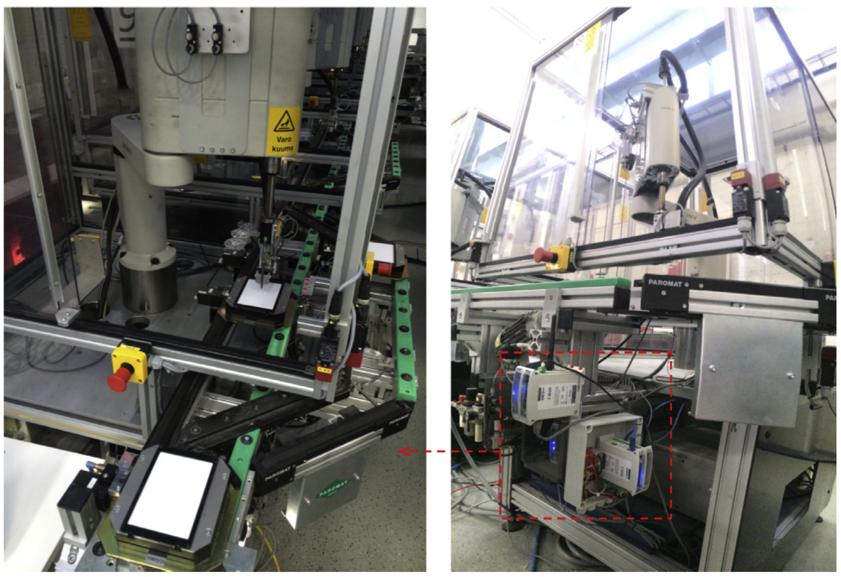


Fig. 3. Three industrial controllers are used for controlling processes of a FASTory line cell.

Table 1
Benchmarking the implementation of the approach in the FASTory line.

	Original	Private local cloud
Production type	Mass	Reconfigurable
Communication protocols	Open and Proprietary	Open
IT infrastructure	Poorly reusable	Highly reusable
Control architecture	Centralized	Distributed
Control logic	Scan based	Event based
Modularity	At manufacturing cell level	At device level
Addressability	Hierarchical	Direct (query)
Processing layer	Fixed	Customizable
Processing of data	At central unit	At device level
Number of computational nodes	1 (centralized architecture)	Many (decentralized architecture)
Plug and play	No	Yes
Output	Signal data	Information
Data format	Heterogeneous	Homogeneous
Message overhead	No	Yes
Message readability by humans	No	Yes
Horizontal and vertical integration	No	Yes
Remote monitoring	Additional middleware is required	From cloud and/or from device
Cyber security	Obscurity (i.e. system isolation)	Defence-in-depth

consumption of the equipment. It should be noted that this research work is not restricted to the use of certain RTUs. For example, some experiments for controlling processes within Raspberry Pi[11] have been already tested [38].

On the other hand, the devices that inhabit in the private cloud need to encapsulate knowledge of the system, which will permit devices to decide and control the production line. In fact, [9,11] describe the use of ontology-based knowledge representation in knowledge-driven approaches in actual production lines.

However, aforementioned research works implement the knowledge localization and manipulation far away from where data is generated (i.e. physical equipment). This research work proposes lowering the knowledge representation to the device level. Actually, an approach for decentralising the knowledge used for knowledge-driven systems within the eScop project solution is described in [40].

The approach discussed in this article goes one step forward because proposes not only the decentralisation and lowering the knowledge to the device level; but also the implementation and control of higher-level functions on devices. In order to benchmark the proposed approach, Table 1 shows different aspects of the original FASTory line configuration that will be improved within further implementation. Some of the aspects have been contrasted with the ones of the research work presented in [2].

As shown in Table 1, the first direct benefit for the FASTory line is the re-configurability of production type. Meanwhile the original line configuration permits the production of one product, the deployment of embedded devices that are capable of controlling and linking different FASTory line resources will permit customizing and ordering remotely products at run time.

On the other hand, the implementation of the cloud brings new features as remote monitoring, addressability of resources, and horizontal and vertical integration. However, the message overhead will be increased due to the fact that the messages being exchanged contain additional information than the signal data created at different locations of the production line. Despite on this, and thanks to the representation of knowledge, the data becomes more readable and understandable for humans. Moreover, as the data format is homogenized the configuration will demand less effort.

Finally, the fact of adding decentralized computational nodes that are linked to physical resources, causes changes to the operation and architecture of the system:

- The control architecture becomes distributed because the nodes will exchange messages in order to solve requests
- Data will be processed at device level, instead of having a central unit for data processing
- The control logic will be event based because nodes will react to notifications related to changes on machine status
- The amount of computational nodes increases and will be scalable. In fact, the implemented SW and HW will be highly reusable and new devices will be deployable at run time
- In contrast to recent approaches in the same production line [12] the knowledge of the system will be decentralized

3.2. Main concepts to be included in device knowledge models

The KB encapsulated in embedded devices is one of the most important element of the proposed architecture because contains the information used by each cloud device to interact and make decisions with the rest of peers. In this research work, such knowledge container is to be implemented within ontologies. More precisely, OWL is the selected language to describe the KB, which will be inspected and updated within SPARQL-based language queries.

Obviously, each KB has different type of information that is based on each device role in the cloud. Nevertheless, there are a set of concepts that must be similarly described in each KB i.e. *Network*, *Service* and *Device*. In this way, the only different data type contained in KBs is the one e.g., used for controlling specific equipment to which the embedded device is connected.

Following Fig. 4, presents the main concepts that each device hosts as local resources in an ontological-based model.

As explained, Fig. 4 illustrates the principal concepts to be described in the ontological model, which is hosted in the embedded devices forming the private cloud. Such kind of information is required to be included in each device because this research work proposes to handle requests and decision-making fully in the private automation cloud itself. Therefore, devices must know the status, location and type of peers that inhabit in the same network. The specific reason for each concept to be included in the ontology is described below:

- *Network:* It contains information about the network or, in this case, the private cloud. It includes the addresses of other devices in the network. In this manner, a device knows other peers' endpoints to send any type of request.
- *Service:* It includes information about the services described in the device as, for example, service and message types and operations. To design such description, OWL-S specification[12] can be used. Through this concept, device capabilities are exposed to other peers, which is a requirement because the functionality description of each device must be accessible in the private cloud. Basically, anything that a device is capable to perform, or knows, must be shared with other devices residing in the same network.
- *Device:* It contains general information of the device as e.g., its type, address, operation system and firmware version. This data is important because a *Device X* can access to *Device Y* basic information. For instance, a *Device X* may employ this information to update its *Network* concept information with the *Device Y* address. Therefore, this supports devices to decide in which situation and how to access other peers.

[11] http://www.raspberrypi.org/.

[12] https://www.w3.org/Submission/OWL-S/.

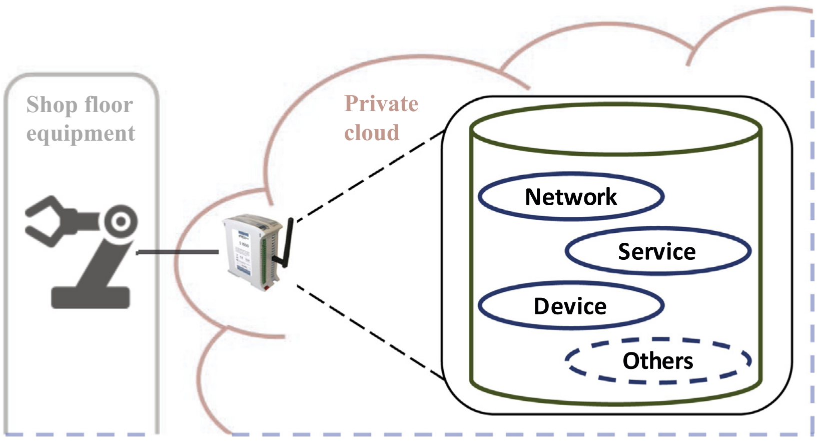


Fig. 4. Main concepts of the ontological-based knowledge representation.

- *Others:* It includes data instances related to the specific role, purpose, operations, etc. that the device is responsible of. Then, *Others* will be a conjunct of classes related to the device type. It should be noted that the *Others* concept of Fig. 4 is a mere representation so that in reality a device can include more than one concept needed for such description in the ontology. For example, if a *Device Z* is controlling an entire cell of an assembly line, the *Others* concept might be implemented as a set of concepts for describing components and processes of that specific cell. On the other hand, if a *Device T* is used for calculating Key Performance Indicators (KPIs), the *Others* concept might consist in a set of concepts for describing the type of data and equations to be calculated.

Basically, the “*other*” concepts included in devices describe information of the specific tasks that each device performs in any controlled system that uses this approach.

Aiming an exemplification of the above described device's ontology main concepts, Fig. 5 shows a possible class hierarchy

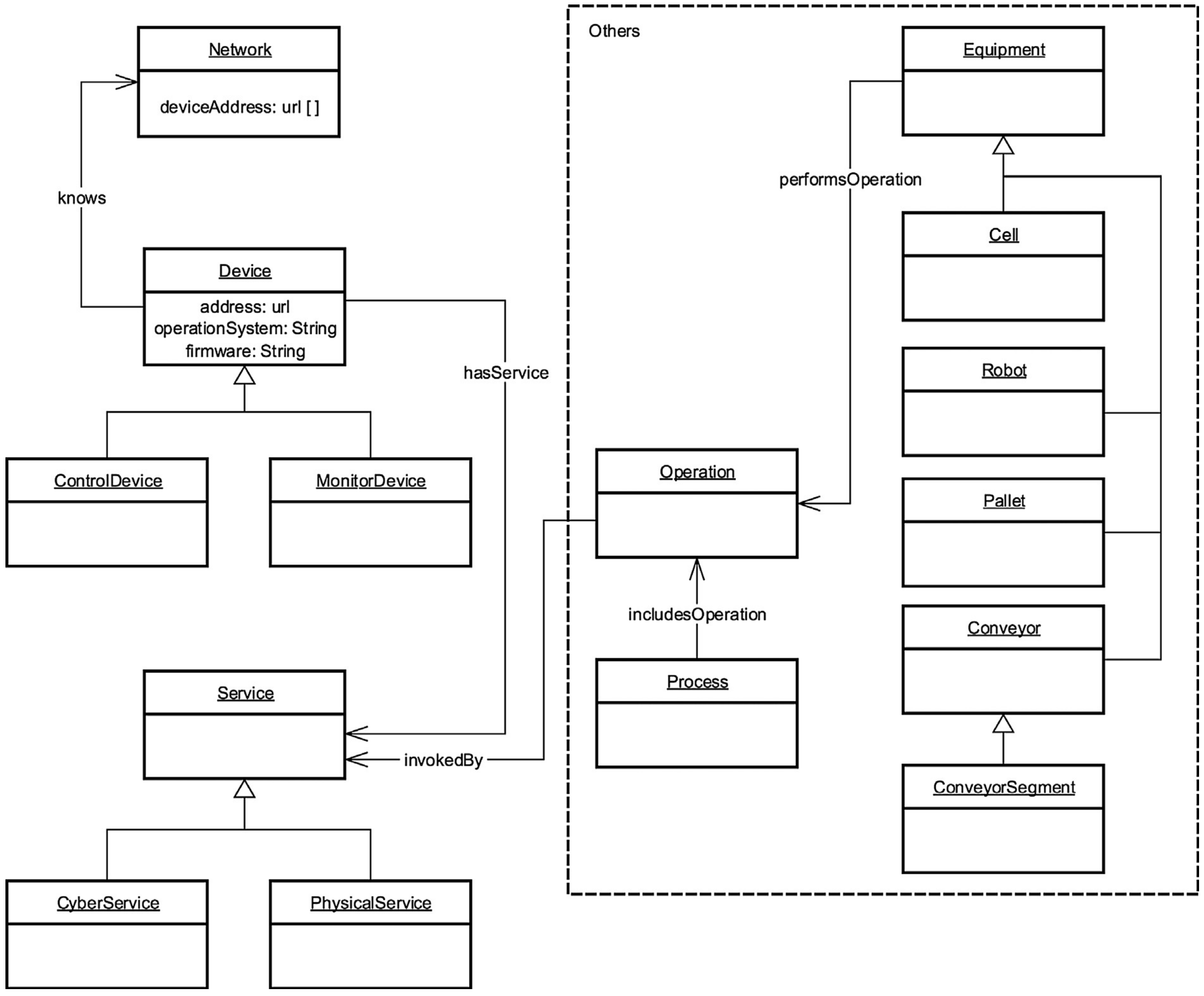


Fig. 5. Class diagram example of a device ontology used for describing and controlling a FASTory line cell.

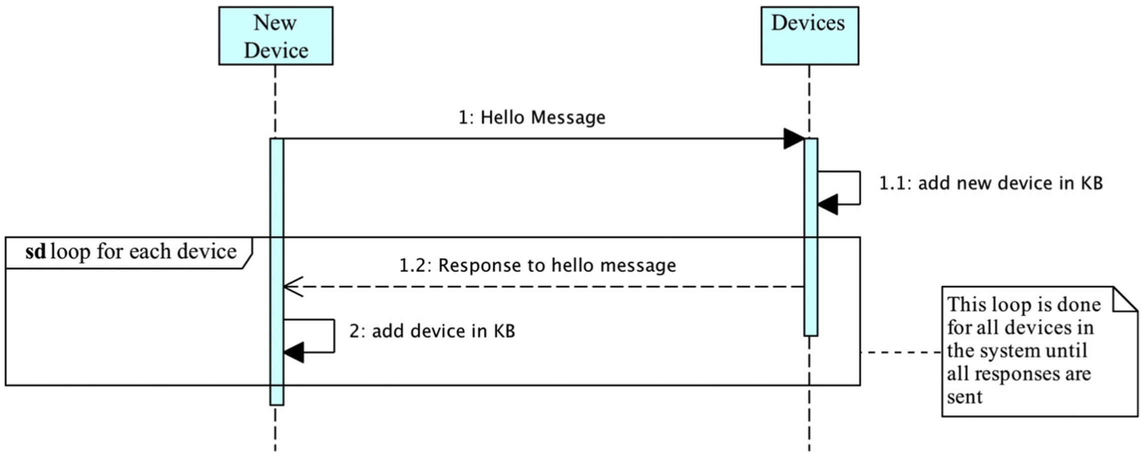

New Device
Devices
1: Hello Message
1.1: add new device in KB
sd loop for each device
1.2: Response to hello message
2: add device in KB
This loop is done for all devices in the system until all responses are sent

Fig. 6. Sequence diagram for registering a new device [38].

of a device, which is used to control a manufacturing cell of the FASTory line. Then, the presented class diagram demonstrates how the concepts can be implemented in a real-case scenario.

First, the presented class diagram shows that the instance of the *Network* class will handle an array, labelled as *deviceAddress*. This attribute is an array which contains all the addresses of the devices of the network. As described in next Section 3.3, once a device is added to the cloud it shares its address so it can be accessed by other devices. Then, each time that a new device enters in the network, the *deviceAddress* array of each device is updated.

On the other hand, the *Device* class is the one to include the device instance. Besides having data type properties, or attributes (as e.g. *address*, *operationSystem* and *firmware*), *Device* is a superclass of *ControlDevice* and *MonitorDevice*, which define the device type. Meanwhile *ControlDevice* type indicates that the device is used for controlling processes with physical consequences in the environment; *MonitorDevice* types includes the device instance if is just used for operations that does not change directly the physical world (e.g., calculations, checking values of sensors and data manipulation). In fact, a device can be classified in both types if it is used for both types of tasks. This classification depends directly on the types of services that the device supports to the private cloud.

Any device's KB of the private cloud includes a description of the services that can be of two types i.e., *CyberService* and *PhysicalService*. Services belonging to *CyberService* are those that does not perform any change in the physical environment as e.g. data request services. On the other hand, services belonging to *PhysicalService* are those which imply physical changes as e.g., move, clamp or draw operations.

Finally, to show how *Others* concept is really implemented in a real-case scenario, Fig. 5 shows a set of classes that are needed for describing the physical equipment and process to execute in FASTory cells. Then, as the device of which the KB's class diagram is presented is responsible of the control of FASTory cell operations must include the description of (i) physical equipment, (ii) operations to execute and (iii) processes to perform. Such information, by order, is included in the ontology within *Equipment* superclass and other subclasses (i.e. *Cell*, *Robot*, *Pallet*, *Conveyor* and *ConveyorSegment*), *Operation* and *Process* classes. Then, certain processes that can be performed in the cell are divided into operations (linked within the *includesOperation* object property) that can be invoked by services described in *Service* class (linked within the *invokedBy* object property). It should be noted that Fig. 5 shows just an example of a device that is controlling operations of a manufacturing cell. Therefore, for a different type of device, the *Others* set of classes will be different. However, *Network*, *Device* and *Service* concepts will have the same structure, following the described approach.

Furthermore, as the presented diagram is just the KB of one device of the private cloud, the KB of the whole cloud would be the integration of all devices' KBs. Then, devices are capable of accessing and sharing these kind of information for accomplishing any distributed reasoning task, which is required for controlling processes. One of the benefits of decentralizing the system knowledge description in the cloud is that the knowledge does not have to reside only in one critical device, which (i) if it fails, the system will fail and (ii) most probably, it will not have enough resources for hosting and managing the model of a large production line. Basically, the entire model is decentralized and hosted by several devices, which cooperate, finding, integrating and reasoning the required information, for solving requests. It should be noted that this knowledge is accessible by other devices using the SPARQL 1.1. Graph Store HTTP Protocol[13].

3.3. Behaviour of embedded devices

Besides the ontological model encapsulation in embedded devices, the behaviour of such devices in the cloud must be also designed and implemented. The exchange of messages between embedded devices will be performed within a set of algorithms that allow devices to cooperate in the cloud. It is absolutely essential that cloud devices include *device initialization* and *reasoning process performance* algorithms, presented in [38]. In order to clarify how these algorithms are to be executed, this section includes a description of each one.

Firstly, the *device initialization* algorithm permits that a new device makes itself discoverable by other peers as shown in the UML diagram of Fig. 6.

Any cloud device will be capable to access to the resources of the new device because the address of new devices is broadcasted to all peers in the cloud in the first step of the *device initialization* algorithm within the "*Hello Message*". Hence, this algorithm must be executed when a device enters in the private cloud. Successively, the new device will receive a response including the address of the corresponding devices that are answering to the "*Hello Message*". In this way, the new device will populate the *Network* class

[13] https://www.w3.org/TR/sparql11-http-rdf-update/.

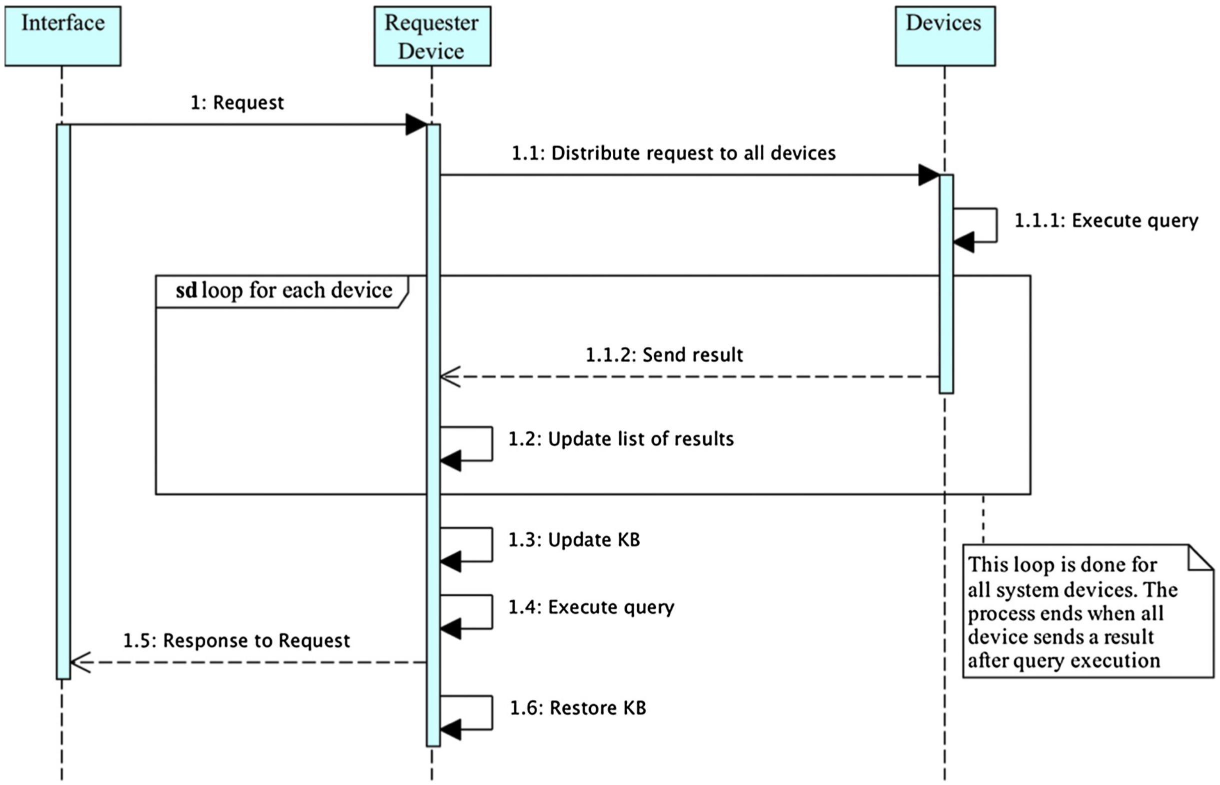

Interface
Requester Device
Devices
1: Request
1.1: Distribute request to all devices
1.1.1: Execute query
sd loop for each device
1.1.2: Send result
1.2: Update list of results
1.3: Update KB
1.4: Execute query
1.5: Response to Request
1.6: Restore KB
This loop is done for all system devices. The process ends when all device sends a result after query execution

Fig. 7. Sequence diagram of a reasoning process performance [38].

of its own KB with the addresses of peers that inhabit in the same cloud.

On the other hand, the *reasoning process performance* algorithm defines the sequence of messages required for exchanging information in a reasoning process as shown in Fig. 7. The reasoning process is required for reasoning the entire system model, which is formed by all resources hosted and managed by the embedded devices.

As it can be seen in Fig. 7, the *reasoning process performance* algorithm permits cloud devices requesting data that is needed for responding to an incoming query. The request is sent by a client to the system through the cloud *interface*. In fact, one of the devices of the cloud will take the lead of the reasoning process. As shown in Fig. 7, this device is known as the *requester device*. The requester device will be selected by the interface according to the available resources that devices have to compute the incoming queries. The task of the requester device is to handle the reasoning process, which implies (i) receiving the request from the interface, (ii) distributing the request to all devices, (iii) collecting the results from devices, (iv) integrating all knowledge, (v) executing the query and (v) sending a response to the interface. Moreover, as it can be seen in the last step of the sequence diagram, the requester device will restore its own KB after handling a reasoning process. This last action is executed in order to remove redundant knowledge that is already hosted by other devices but needed in the knowledge integration.

The reasoning *process performance algorithm* is a process that requires the cooperation of all devices for integrating all the knowledge related to the request and creating a valid response that must satisfy the client. The incoming queries will request (i) information for monitoring resources, (ii) computation of cloud resources or (iii) order a product that can be performed by the manufacturing system being controlled by the private cloud. Therefore, solving requests will frequently imply the execution of services hosted by different embedded devices.

3.4. Implementing ontologies for embedded devices and exemplifying distributed reasoning in the FASTory line

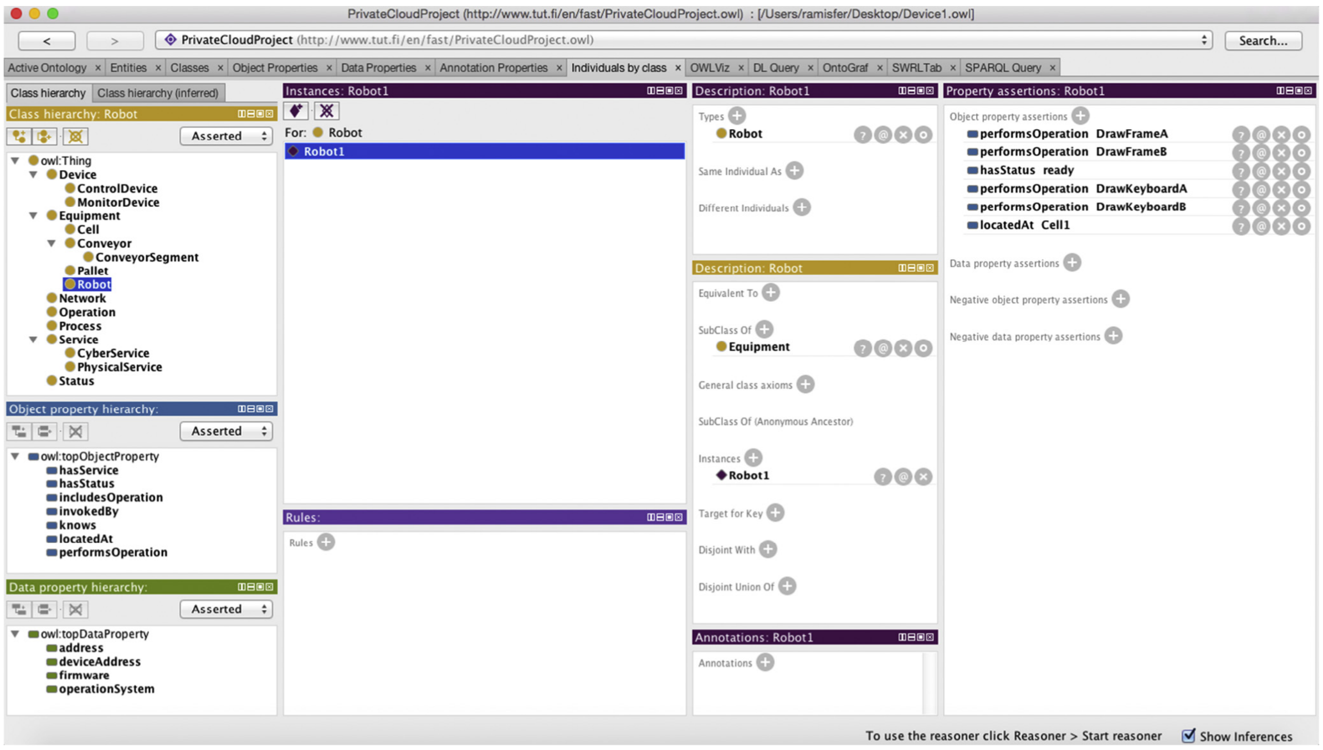
The objective of this section is to show an example of executing distributed reasoning in a concrete manufacturing scenario, i.e. the FASTory line.

Firstly, an ontological model that follows the class hierarchy shown in Fig. 5 is created within an ontology editor. For this experiment, Protégé[14] has been used as the tool for editing the model. Once the ontology is created, it is populated with the instances being controlled by corresponding embedded devices. Thereby, if the *device 1* is hosting *model 1*, the instances of *model 1* will represent the industrial components being controlled by *device 1*. Fig. 8 shows the ontology of the device that is connected and controls the components of the FASTory *cell 1*.

As it can be seen in Fig. 8, the ontology includes all the classes shown in the ontology model presented in Fig. 5. In addition, Fig. 8 shows, as an example, that the *Robot 1* (controlled by the *device 1*) has a set of operations that can be performed: *DrawFrameA, DrawFrameB, DrawKeyboardA* and *DrawKeyboardB*. These operations will be executed when the linked services through the *hasService* object property are invoked.

After the design and implementation of each device ontology, the models are deployed into corresponding devices. The deployment of device models will differ according to the type of embedded devices, which will employ different platforms. Nevertheless, this task will imply always two major steps: (i) to place the

[14] http://protege.stanford.edu/.



```
owl:Thing
    Device
        ControlDevice
        MonitorDevice
    Equipment
        Cell
        Conveyor
            ConveyorSegment
        Pallet
        Robot
    Network
    Operation
    Process
    Service
        CyberService
        PhysicalService
    Status
```

Types: Robot

SubClass Of: Equipment

Instances: Robot1

```
performsOperation  DrawFrameA
performsOperation  DrawFrameB
hasStatus  ready
performsOperation  DrawKeyboardA
performsOperation  DrawKeyboardB
locatedAt  Cell1
```

```
owl:topObjectProperty
    hasService
    hasStatus
    includesOperation
    invokedBy
    knows
    locatedAt
    performsOperation
```

```
owl:topDataProperty
    address
    deviceAddress
    firmware
    operationSystem
```

Fig. 8. OWL model hosted by device 1.

```
PREFIX dr:<http://www.tut.fi/en/fast/PrivateCloudProject.owl#>
SELECT ?Robot ?RobotStatus ?Cell ?Operation
WHERE {
?Robot dr:hasStatus ?RobotStatus.
?Robot dr:locatedAt ?Cell.
?Robot dr:performsOperation ?Operation.
}
```

Fig. 9. Example of SPARQL query to be solved by the private cloud within the distributed reasoning process.

ontology file into the device and (ii) to implement an interface that permits the interaction with the model. For example, the ontology model can be placed into the SD card of a Raspberry Pi and the interface can be implemented within Apache Jena,[15] which is a framework that permits the management of RDF models. In fact, the *Apache Jena Fuseki* is a server that can run as an operating system service providing the SPARQL 1.1. and the SPARQL 1.1. Graph Store HTTP protocols to query, store and update RDF-based models.

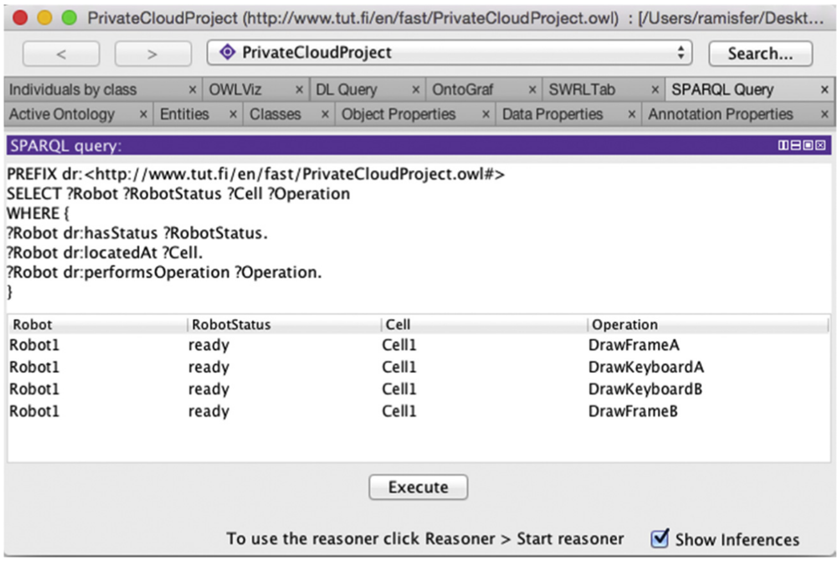
For the FASTory line, incoming queries to the system can be simple monitoring requests, such as asking the status of a robot or complex requests, e.g., order the production of a mobile phone. In any case, the system will react in the same way to requests, i.e. executing the distributed reasoning process. Fundamentally, the execution of the aforementioned algorithm permits the integration of knowledge that is located at different computational nodes to ensure that the validation of the query is done according to the whole model information.

As the research proposes the use of RDF-based models, the ontology has been implemented in OWL syntax, which can be queried within SPARQL queries. In order to present an example of distributed reasoning in the FASTory line, Fig. 9 shows a monitoring query to be sent and executed by the private automation cloud.

The query shown in Fig. 9 can be used for requesting a list of robots including their status, location and operations that can perform. As previously said, each embedded device of the FASTory line hosts an ontology with similar class hierarchy but populated with the specific industrial equipment that each peer controls. As each device is used for controlling a different cell, a unique device is not capable of answering the incoming query shown in Fig. 9 including the status of all assembly line robots. Basically, as an example, *device 1* will be capable of answering with the status, location and operations of *Robot 1* but not with the ones of, e.g., *Robot 2*, as shown in Fig. 10.

Nevertheless, the execution of the distributed reasoning process will permit the cloud to achieve a valid answer to be returned to the cloud client. As it can be seen in the sequence diagram of Fig. 7, the incoming query will be broadcasted from the requester device to all cloud devices. Then, each device having any knowledge instance of the type of any query element will be returned to the requester device. Afterwards, the requester device will execute the query with all collected knowledge and conclude the answer, which will be redirected to the cloud interface. Assuming that (i) the integration model is created after the execution of the

[15] https://jena.apache.org.



```
PREFIX dr:<http://www.tut.fi/en/fast/PrivateCloudProject.owl#>
SELECT ?Robot ?RobotStatus ?Cell ?Operation
WHERE {
?Robot dr:hasStatus ?RobotStatus.
?Robot dr:locatedAt ?Cell.
?Robot dr:performsOperation ?Operation.
}
```

Robot	RobotStatus	Cell	Operation
Robot1	ready	Cell1	DrawFrameA
Robot1	ready	Cell1	DrawKeyboardA
Robot1	ready	Cell1	DrawKeyboardB
Robot1	ready	Cell1	DrawFrameB

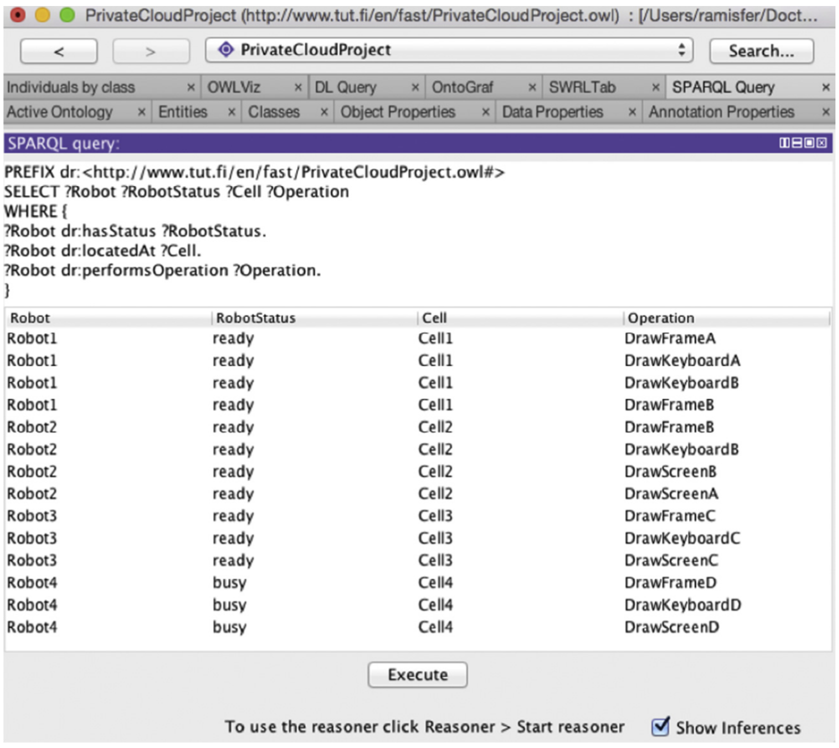
Fig. 10. Result of query execution in *device 1*.

distributed reasoning process and that (ii) this is tested in a reduced version of the FASTory line with 4 cells: the requester device for the query shown in Fig. 9 obtain the result depicted in Fig. 11.

As it can be seen in the result depicted by Fig. 11, the requester device will be capable of answering the incoming query including all robot status, location and operations of the line. The result is achieved due to previous integration of each device knowledge related to the incoming query.

Another example to discuss the execution of the distributed reasoning process can be a request for routing a pallet through the assembly line. In order to control the transport of pallets in the FASTory line, services describing conveyor operations must be executed in certain order. For example, assuming that the client requests for a valid path to route the pallet from *point A* to *point B*, the cloud needs first to conclude the layout of the line. Then, assuming that those points are manufacturing cells and that the query asks about locations, all devices will share the knowledge instances related to location type; i.e., robots, conveyors and manufacturing cells. Then, the requester device will be capable of shaping the line because the integrated model will contain all cell physical connections.

In fact, when an incoming query requests the production of a mobile phone, the locations of each robot that can perform required operations for the order must be located. As shown in Fig. 11, the current FASTory line configuration has no single cell



```
PREFIX dr:<http://www.tut.fi/en/fast/PrivateCloudProject.owl#>
SELECT ?Robot ?RobotStatus ?Cell ?Operation
WHERE {
?Robot dr:hasStatus ?RobotStatus.
?Robot dr:locatedAt ?Cell.
?Robot dr:performsOperation ?Operation.
}
```

Robot	RobotStatus	Cell	Operation
Robot1	ready	Cell1	DrawFrameA
Robot1	ready	Cell1	DrawKeyboardA
Robot1	ready	Cell1	DrawKeyboardB
Robot1	ready	Cell1	DrawFrameB
Robot2	ready	Cell2	DrawFrameB
Robot2	ready	Cell2	DrawKeyboardB
Robot2	ready	Cell2	DrawScreenB
Robot2	ready	Cell2	DrawScreenA
Robot3	ready	Cell3	DrawFrameC
Robot3	ready	Cell3	DrawKeyboardC
Robot3	ready	Cell3	DrawScreenC
Robot4	busy	Cell4	DrawFrameD
Robot4	busy	Cell4	DrawKeyboardD
Robot4	busy	Cell4	DrawScreenD

Fig. 11. Query result after distributed reasoning process performance.

capable of assembling all possible component types of a mobile phone (i.e. same type of keyboard, screen and frame). Then, the execution of required operations implies to conclude first the location and status of the equipment capable of performing such operations.

4. Discussion

Once the principles of the private cloud architecture and requirements of embedded devices have been described, potential and challenges of the approach are presented in this section.

4.1. Potential of the approach

The proposed research looks forward the creation of a private automation cloud formed by embedded devices that handle functionality and knowledge management, nowadays done centrally and away from where the data is generated (e.g., industrial equipment, machines, sensors, actuators). Moreover, the presented approach proposes to create a solution within resource-constrained devices, which are capable to cooperate for performing distributed reasoning. The main potential of the presented approach is:

- *Network:* The implementation of such approach will be useful for evaluating the limits of using embedded devices not only for data collection and shop floor process control, but also i) to process the data and transform it into usable information and ii) to cooperate with other peers in the cloud for performing the required distributed reasoning, based on demand of private cloud resources. Hence, the development of the proposed distributed system will be useful to test that the new generation of embedded devices are ready to operate and fulfil the requirements of industrial systems, which are complex and dynamic environments. In addition, this research work has a direct impact on future awareness, flexibility and re-configurability of manufacturing systems [41] in the monitoring and supervisory control area.
- *Reduction of high integration cost:* One of the motivations of this research is the high-cost of integration in the industrial automation field, which can be reduced by the use of distributed intelligence [42]. Then, this research can support such reduction by the automation of the embedded systems configurability in factory floors through reasoning, using the whole range of computational resources available in the proposed cloud.
- *Decentralisation of knowledge-driven management solutions:* This approach follows some principles described in novel knowledge-driven approaches, which are presented in [40] as an alternative for conventional architecture and operation of MES [11]. The CPS integration permits the development of the proposed cloud that control system processes. Potentially, this fact directly contributes to improvement of efficiency and productivity of industrial automation processes.
- *Integration with external systems:* The industry is already employing internet technologies and web-based standards for semantic modelling, linking data and requesting resources. This allows entities to describe, request and manipulate system's KB for decision-making and execution of operations. As such technologies and standards are the ones to be used in presented approach, the interoperability of the cloud with actual external systems (e.g. legacy systems) is achievable. Then, if a third party is interested in demanding computational resources to the cloud, it can do it remotely and without communication issues but, obviously, with prior agreement between the cloud owner and interested organisations.
- *Multi-cloud or interconnection of clouds:* A multi-cloud collaboration between similar clouds owned by the same organization at different geolocations is possible. This connection can be used for sharing and, hence, increasing computational resources of a unique cloud. In this way, different clouds can be specialized in different computation of data, besides controlling the processes of connected equipment. Actually, clouds that are smaller in terms of computation power can consume e.g. data processed with complex and expensive algorithms. Then, scheduling plans, datasets, KPI calculations, among other sort of data can be requested and used between different clouds that manages similar processes in a same organization.

4.2. Challenges of the approach

For making possible the realization of the presented approach, some challenges become, at the end, requirements that must be addressed. Karnouskos et al. describes in [43] their vision for cloud-based industrial CPS, presenting trends and challenges on this fairly new research topic. In fact, some of the already presented challenges in [43] are also applicable in this research. The main challenges of the presented approach are:

- *Implementation of device behaviour:* Explained algorithms in Section 3.3 and others e.g., dynamic discovery of devices must be implemented and evaluated. The importance of these algorithms relies on the need of developing robust and efficient device intercommunication in the cloud.
- *Development of messages:* Previous challenge implies the need of designing message types and structures for information exchange. This point is important so that devices can interact between themselves or/and external users of the cloud. These messages can be understood as queries or request that are different depending on the situation in which devices will be involved. Then, besides the technology or standard within messages are written, different types of messages and corresponding content must be defined. In fact, a possible outcome of this challenge is the creation of a communication protocol for devices.
- *Horizontal scalability:* One of the features of CC the horizontal scalability i.e. increasing cloud capacity by connecting multiple entities so that all entities work as a single one. Although this research work contemplates the incorporation of similar devices with different roles in the system, the main idea of the private automation cloud is the cooperation of devices to work as a unique system. Simply, when a device is added to the cloud, it includes additional resources to be employed by the rest of devices. Therefore, from this perspective, the horizontal scalability is a challenge of the approach.
- *Fault tolerance:* Communications are vulnerable to faults. Thus, in cloud-based systems, *fault tolerance* is an aspect to take into consideration for ensuring that services are available and accessible. In order to guarantee a correct execution of cloud services, specific mechanisms must be provided to solve or, in best case scenario, anticipate to faults in the system. Conceptually, this challenge is about addressing robustness and reliability of distributed system communications.
- *Orchestration and/or choreography service execution:* Independently on device behaviour, cloud devices might need the incorporation of software that will allow the orchestration or choreography for setting the order of service operation execution. In addition, this challenge implies the development of a goal-oriented mechanism for integration of computational resources because service operations of a same process can be encapsulated in different embedded devices.

- *Cloud security:* The private cloud to be implemented will require an investment on data security. Fundamentally, the cloud is meant to be private. Then, it must be accessible only by the allowed organization, users and/or third party applications.
- *Others:* Some challenges addressed in [43] that must be considered as: cross-layer collaboration by CPS, migration and impact of CPS to existing industrial automation approaches, and semantic-driven interaction.

5. Conclusion

Recent developments in CPS permit the integration of semantic technologies and embedded devices, which propound a new scenario with many potential and challenges. Thus, this article presents the principles, challenges and potential of a private local automation cloud that is built by CPS. This research considers and rethinks the automation pyramid shown in ISA-95 and other central knowledge management solutions because the process of system information is done at the device level. Thus, one possible outcome of the presented approach is the redefinition of the business model for actual industrial systems.

Advances on CPS in recent research works and its impact on recent industrial system are presented in [11,44]. First, the employment of ontological-based knowledge representation as e.g. presented in early approach description of the eScop project in [11], permits the knowledge-driven solutions as a promising solution in the field. On the other hand [44], presents directly recent advances and trends of CPS in industrial informatics. Nevertheless, aforementioned research works do not consider that embedded devices can take the role of processing information and making decisions for production lines process control, which is the main objective of the presented approach.

Then, this article presents an alternative approach for the use of CPS for building a cloud that can implement other research works solutions but in the location where data is generated. Possibly, this approach will decrease industrial system integration time and costs because it will reduce cross-layer communication. In addition, the article presents an opportunity for evaluating the limitations of using a private cloud composed by embedded devices in the industrial automation domain.

Further work will include the development of models, algorithms and experimental cases for evaluating the cloud performance. Indeed, the challenges that are described in Section 4.2 must be taken into account when developing the proposed cloud. Then, this article can be used as a high-level roadmap for described development.

Acknowledgements

The authors gratefully acknowledge the support of the graduate school funding of Tampere University of Technology in carrying out this work.

References

[1] M. Qusay H., "SOA and Web Services", 2005. [Online]. Available: <http://www.oracle.com/technetwork/articles/javase/soa-142870.html> (accessed: 02-March-2016).
[2] L.E.G. Moctezuma, J. Jokinen, C. Postelnicu, J.L.M. Lastra, Retrofitting a factory automation system to address market needs and societal changes, in: 2012 10th IEEE International Conference on Industrial Informatics (INDIN), 2012, pp. 413–418.
[3] I. International Electrotechnical Commission, "IEC 61131-3:2013 | IEC Webstore," 2013. [Online]. Available: <https://webstore.iec.ch/publication/4552> (accessed: 02-March-2016).
[4] I. International Electrotechnical Commission, "IEC 61499-1:2012 | IEC Webstore", 2012. [Online]. Available: <https://webstore.iec.ch/publication/5506> (accessed: 02-March-2016).
[5] V. Vyatkin, IEC 61499 as enabler of distributed and intelligent automation: state-of-the-art review, IEEE Trans. Ind. Inform. 7 (4) (2011) 768–781.
[6] C. Popescu, J.L.M. Lastra, On ontology mapping in factory automation domain, in: IEEE Conference on Emerging Technologies and Factory Automation, 2007, ETFA, 2007, pp. 288–292.
[7] P. Novák, E. Serral, R. Mordinyi, R. Šindelář, Integrating heterogeneous engineering knowledge and tools for efficient industrial simulation model support, Adv. Eng. Inform. 29 (3) (2015) 575–590.
[8] J. Puttonen, A. Lobov, J.L.M. Lastra, Maintaining a dynamic view of semantic web services representing factory automation systems, in: 2013 IEEE 20th International Conference on Web Services (ICWS), 2013, pp. 419–426.
[9] B. Ramis et al., Knowledge-based web service integration for industrial automation, in: 2014 12th IEEE International Conference on Industrial Informatics (INDIN), 2014, pp. 733–739.
[10] L. Petnga, M. Austin, An ontological framework for knowledge modeling and decision support in cyber-physical systems, Adv. Eng. Inform. 30 (1) (2016) 77–94.
[11] G. Marco, Open Automation of Manufacturing Systems through Integration of Ontology and Web Services, 2013, pp. 198–203.
[12] S. Iarovyi, W.M. Mohammed, A. Lobov, B.R. Ferrer, J.L.M. Lastra, Cyber-physical systems for open-knowledge-driven manufacturing execution systems, Proc. IEEE PP (99) (2016) 1–13.
[13] E.A. Lee, Cyber physical systems: design challenges, in: 2008 11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing (ISORC), 2008, pp. 363–369.
[14] S.M. Hashemi, A.K. Bardsiri, Cloud computing vs. grid computing, ARPN J. Syst. Softw. 2 (5) (2012) 188–194.
[15] D.K. Barry, Web Services, Service-Oriented Architectures, and Cloud Computing: The Savvy Manager's Guide, Newnes, 2012.
[16] K. Keahey, M. Tsugawa, A. Matsunaga, J. Fortes, Sky computing, IEEE Internet Comput. 13 (5) (2009) 43–51.
[17] G. Lawton, Developing software online with platform-as-a-service technology, Computer 41 (6) (2008) 13–15.
[18] S. Goyal, Public vs private vs hybrid vs community – cloud computing: a critical review, Int. J. Comput. Netw. Inf. Secur. 6 (3) (2014) 20–29.
[19] A.W. Colombo et al. (Eds.), Industrial Cloud-Based Cyber-Physical Systems, Springer International Publishing, Cham, 2014.
[20] I.M. Delamer, J.L.M. Lastra, Loosely-coupled automation systems using device-level SOA, 2007 5th IEEE International Conference on Industrial Informatics, vol. 2, 2007, pp. 743–748.
[21] A.W. Colombo, S. Karnouskos, T. Bangemann, Towards the next generation of industrial cyber-physical systems, in: A.W. Colombo, T. Bangemann, S. Karnouskos, J. Delsing, P. Stluka, R. Harrison, F. Jammes, J.L. Lastra (Eds.), Industrial Cloud-Based Cyber-Physical Systems, Springer International Publishing, Cham, 2014, pp. 1–22.
[22] A.J.C. Trappey, C.V. Trappey, U. Hareesh Govindarajan, A.C. Chuang, J.J. Sun, A Review of Essential Standards and Patent Landscapes for the Internet of Things: A Key Enabler for Industry 4.0, Adv. Eng. Inform.
[23] J. Lee, B. Bagheri, H.-A. Kao, A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems, Manuf. Lett. 3 (2015) 18–23.
[24] R.J. Brachman, H.J. Levesque, Knowledge Representation and Reasoning, Morgan Kaufmann, 2004.
[25] V. Premkumar, S. Krishnamurty, J.C. Wileden, I.R. Grosse, A semantic knowledge management system for laminated composites, Adv. Eng. Inform. 28 (1) (2014) 91–101.
[26] F. Song, G. Zacharewicz, D. Chen, An ontology-driven framework towards building enterprise semantic information layer, Adv. Eng. Inform. 27 (1) (2013) 38–50.
[27] A. Lobov, F.U. Lopez, V.V. Herrera, J. Puttonen, J.L.M. Lastra, Semantic Web Services framework for manufacturing industries, in: IEEE International Conference on Robotics and Biomimetics, 2008, ROBIO 2008, 2009, pp. 2104–2108.
[28] W. Lu et al., Selecting a semantic similarity measure for concepts in two different CAD model data ontologies, Adv. Eng. Inform. 30 (3) (2016) 449–466.
[29] H. Liu, M. Lu, M. Al-Hussein, Ontology-based semantic approach for construction-oriented quantity take-off from BIM models in the light-frame building industry, Adv. Eng. Inform. 30 (2) (2016) 190–207.
[30] T.R. Gruber, A translation approach to portable ontology specifications, Knowl. Acquis. 5 (2) (1993) 199–220.
[31] J.L.M. Lastra, I.M. Delamer, F. Ubis, Domain Ontologies for Reasoning Machines in Factory Automation, ISA, 2010.
[32] E. Negri, L. Fumagalli, M. Garetti, L. Tanca, A review of semantic languages for the conceptual modelling of the manufacturing domain, in: Proceedings of the XIX Summer School Francesco Turco, 2014, Senigallia, Ancona, 2014, pp. 1–8.
[33] J. Puttonen, A. Lobov, M.A. Cavia Soto, J.L. Martinez Lastra, Planning-based semantic web service composition in factory automation, Adv. Eng. Inform. 29 (4) (2015) 1041–1054.
[34] S. Abburu, A survey on ontology reasoners and comparison, Int. J. Comput. Appl. 57 (17) (2012).
[35] B. Ramis Ferrer, B. Ahmad, D. Vera, A. Lobov, R. Harrison, J.L. Martínez Lastra, Product, process and resource model coupling for knowledge-driven assembly automation, Autom. 64 (3) (2016).
[36] J. Puttonen, A. Lobov, M.A.C. Soto, J.L.M. Lastra, A Semantic Web Services-based approach for production systems control, Adv. Eng. Inform. 24 (3) (2010) 285–299.

[37] Y. Evchina, J.L. Martinez Lastra, Hybrid approach for selective delivery of information streams in data-intensive monitoring systems, Adv. Eng. Inform. 30 (3) (2016) 537–552.

[38] B.R. Ferrer, S. Iarovyi, L. Gonzalez, A. Lobov, J.L.M. Lastra, Management of distributed knowledge encapsulated in embedded devices, Int. J. Prod. Res. (2015) 1–18.

[39] M.J.A.G. Izaguirre, A. Lobov, J.L.M. Lastra, OPC-UA and DPWS interoperability for factory floor monitoring using complex event processing, in: 2011 9th IEEE International Conference on Industrial Informatics (INDIN), 2011, pp. 205–211.

[40] Borja Ramis Ferrer, A proposal of decentralized architecture for OKD-MES, in: S. Strzelczak, P. Balda, M. Garetti, A. Lobov (Eds.), Open Knowledge-Driven Manufacturing & Logistics, the ESCOP Approach, Warsaw University of Technology Publishing House, Warsaw, 2015, pp. 331–340.

[41] V.V. Vyatkin, J.H. Christensen, J.L.M. Lastra, OOONEIDA: an open, object-oriented knowledge economy for intelligent industrial automation, IEEE Trans. Ind. Inform. 1 (1) (2005) 4–17.

[42] I.M. Delamer, J.L.M. Lastra, Service-oriented architecture for distributed publish/subscribe middleware in electronics production, IEEE Trans. Ind. Inform. 2 (4) (2006) 281–294.

[43] S. Karnouskos, A.W. Colombo, T. Bangemann, Trends and challenges for cloud-based industrial cyber-physical systems, in: A.W. Colombo, T. Bangemann, S. Karnouskos, J. Delsing, P. Stluka, R. Harrison, F. Jammes, J.L. Lastra (Eds.), Industrial Cloud-Based Cyber-Physical Systems, Springer International Publishing, Cham, 2014, pp. 231–240.

[44] B.B. Jay Lee, A cyber-physical systems architecture for industry 40-based manufacturing systems, SME Manuf. Lett. (2014).