Management of IoT Devices in a Physical Network

Jose Ferreira
Centre of Technology and Systems, CTS, UNINOVA
Campus da Caparica, 2829-516 Caparica, Portugal
japf@uninova.pt

Joao Nuno Soares
UNL, Faculdade de Ciencias e Tecnologia
Campus da Caparica, 2829-516 Caparica, Portugal
jn.soares@campus.fct.unl.pt

Ricardo Jardim-Goncalves
Centre of Technology and Systems, CTS, UNINOVA
Campus da Caparica, 2829-516 Caparica, Portugal
rg@uninova.pt

Carlos Agostinho
Centre of Technology and Systems, CTS, UNINOVA
Campus da Caparica, 2829-516 Caparica, Portugal
ca@uninova.pt

***Abstract*— The internet of things is an evolving trend, connecting "things" to the internet, exchanging and collecting information about the surrounding environment. This is possible, by using different devices such as sensors and actuators feeding information, which when connected together enable to monitor and interact with the surrounding environment. Nowadays, there are many devices with different functionalities from multiple manufacturers, which create an interoperability issue of how to inter-connect and manage this heterogeneous world of devices. In order to promote interoperability and effectively manage devices it is required to store their data and at the same time handle the registration of each device in a network. The development of a reference architecture can provide a solution for this problem, however a consensus on a reference architecture that can effectively manage this heterogeneous world of devices, is still not yet achieved. Device management is a key feature for any real world solution for a scalable reliable Interoperable cloud platform. Adding new devices to a network and assign the necessary services, interfaces and rules will provide a fast track not only in terms of a business-oriented solution but also in back-end management and interoperability. This paper proposes a device management solution based on an architecture and model, to support the registration of devices and propose the necessary services and events, which can be used by a specific device, during its life span inside the physical network.**

Keywords — Internet of Things; Interoperability; Intelligent Services; Device Management; Complex Event Processing

I. INTRODUCTION

The Internet of Things (IoT) is not a new topic but an evolving trend. The continuous advances in technology have allowed a large number of devices with communication and computing capabilities to interact with the surrounding environment at a low cost. In this context smart home, e-health, automation, industrial manufacturing, intelligent transportation of people and goods are only a few examples of possible application scenarios where the impact will be significant. The IoT can potentially transform the business and consumer markets and change the way we all live and work, locally and globally. Many industries are facing difficulties regarding the challenge of building highly scalable, secure and data management infrastructures that run on standard protocols, in order to improve productivity and efficiency to remain competitive. This transformation will create new markets and new business opportunities, not only to provide solutions for demanding industries but also to the average consumer [1].

Many projects have targeted the problem of creating frameworks to solve the issue of interconnecting heterogeneous devices, although aiming at specific application targets [2]. This creates a fragmentation problem, for every application a completely new solution is created, which arises a few concerns. Among these concerns, are loss in productivity and increase in expenses of service providers for addressing problems with computers, mobile phones, broadband service, devices, and other issues, therefore consumers and businesses stand to lose plenty when it comes to technical problems. The problem is even more overwhelming considering the exponential number of devices. Ericsson estimates that by 2020 there will be 28 million connected devices from which 15 billion will be from IoT devices [3]. The increasing number of devices creates difficulties when cooperation is required with each other, since each one uses different communication protocols, models and technologies. The advantage of having different devices interconnected for businesses, feeding information about an environment, also comes with complications, since the need for new and more efficient ways to manage the devices and their data is a continuous requirement, where connected systems are able to "talk the same language". This arise the issue of how to inter-connect and manage a fragmented world of solutions.

The path to achieve cooperation and interoperability starts with establishing and ensuring a common understanding of the domain [4]. A step forward, in this direction, is through device management supported by a reference architecture. By handling the entry of a device into a network it is possible to know what type of device is being used, the communication protocol, promoting interoperability that, when reached, allows any IoT device to be able to connect to any other device or system and exchange information anytime, anywhere [5]. A need to create a system arises to promote a device management solution that works smoothly in a sustainable interoperability environment, adapting to changes that are occurring in a network [6].

This paper proposes a solution for attaching devices to a network and provides relevant information based on a model, an ontology and events to manage and facilitate interoperability

of a large number of connected devices. The addition of a new device is accomplished via a service that according to the ontology attribute the necessary generic services and events for device management. The next section describes the concepts of IoT, device management, relevant reference projects and technologies. Section 3 the architecture and model for supporting data collection and device management is presented, describing a methodology and its module. Section 4 presents a scenario in a manufacturing environment and finally in section 5 the conclusions and future remarks.

II. Baseline for IoT Interoperability of Device Network

The vision of IoT is supported by the ability to provide services originated from a large number of heterogeneous objects to deliver relevant information from the surrounding physical world and enable a continuous interaction [7]. With the fast growth that is being verified over the years, several problems have been identified, creating the need to mature it, opening new doors to evolve over time. Some problems are related with the proprietary device/systems in use by different enterprises, increasing the level of difficulty to be solved or bypassed with the support of other solutions. Since the proprietaries see an advantaged by using a proprietary ecosystem, they tend to create devices/systems that are only compatible with their IoT products [8].

These situations, open doors for innovation in IoT, in this work a way is explored to support an homogeneous network, which allows different devices/systems to cooperate between them. However, several problems arise and need to be dealt with, for example issues related with the different communication protocols (e.g. WiFi, Bluetooth, etc.) and based on different information models, making communication between them, difficult to achieve. Another problem, due to the different protocols in the data layer, is creating a semantic difficulties regarding interoperability when exchanging data with a different device is required. In the situation of a system being able to manage different devices, by knowing the communication protocol in use, and the structure of the exchanging data, an interoperable device network is reached.

A. Factory Virtualization with Support of CPS System

The increase of number of sensors is bringing benefits to the industrial area, with the use of new technologies, enabling a synchronization of the digital and real world, providing real-time access to sensorial information. At the same time, industries gain an advanced networking and processing capabilities to actively cooperate within a factory environment [9], [10]. This advantage is reached through the use of enterprise resources (as materials, devices, people, etc.) as data information, where some applications that exploit the retrieved knowledge need to understand all relations and inter-dependencies between each of the resources, forming a Cyber Physical Systems (CPS). To accomplish this, a need to virtualization a resources arises.

Virtualization is a term that refers to the abstraction of computer resources in order to improve usage. This abstraction represents a real-world device, consisting of a set of commands to enable the device to receive and send data. By doing so, the information provided by the device can be enriched in a unified integrated operating platform, for users and applications, based on aggregation of diverse and autonomous resources [11].

When virtualizing factory resources, the enterprise removes the complexity in their internal operations, creating an environment which improves agility, responsiveness, and a decentralized decision-making [12]. Using a virtualization of a simple resource allows to abstract, model and simulates the full automation of a process, which represents a real physical entity [13]. The increase availability and accessibility of sensors, data acquisition systems and computer networks, together with the competitive nature of the industrial sector makes more and more enterprises to implement these technologies in their manufacturing processes and products. This leads to an increase of industrial enterprises that introduce IoT and CPS technologies starting with embedding sensors in manufacturing equipment and ending in a complete interoperable, scalable and secure solution. Taking this into consideration, this big use of sensors and networked machines, results in continuous generation of high volumes of data, also known as Big Data [14]. By integrating CPS within the productions, logistics and services in their industrial practices, it will transform today's factories, with a significant economic potential [15]. Since, each enterprise implements the sensor network, as their own, requiring architectures and guidelines to support them, providing advanced connectivity that need to ensure a real-time data acquisition from the physical world with information feedback [16].

B. Architecture & Reference Models for IoT Networks

IoT ready networks require different communication and processing models. Today, there is not a standard way of understanding or describing these models for an IoT system. By using a reference model to provide clear definitions and descriptions enables an IoT system to be simple, organized and standard. In order to implement or use a reference model, an architecture is also defined to provide the building blocks that promote features like: scalability; interoperability; reliability; security and privacy; quality of service [17].

One of the main goals of both architecture and reference model is to monitor and manage a device network. Therefore there is a need to provide a notion of the device in use (characteristics, functionalities, etc.) and its configuration (to know what the device is contribution to the network) in the network. In literature several ways to support device management exists, hence this section provides a study to identify the solution that best suits the problem of this work. Research projects goals usually encapsulate a more complete solution while research papers address more specific issues.

There are many projects that address this issue, for this work, a few were selected, based on level of implementation and development of their proposed works, as well as, relevance for the works presented in this thesis. According to Li et al. in [18], an IoT architecture should have the following characteristics: **Interoperability** when working with the different standards, devices and protocols; **Dynamic adaptation** means the adaptive capability to respond to

unexpected changes, where different devices may have different reactions; **Scalability** indicates the capability to accommodate an increase number of entities, to process a growing volume of work, and/or to handle a larger scale; **Security and Privacy**, a lot of sensitive and private information is used, hence security and privacy functionalities to detect and monitor anomalies unauthorized accesses; **Device Management & Service Discovery**, the architecture needs to be prepared to support the management of devices and propose methodology for discovery new devices in the network; **Context Awareness** explains how context is expressed and formalized. The increase of the abstraction level can improve the ability of reading, understanding and using context; **Storage & Big Data**, the management of large volumes of data and the capacity to handle big data. Table 1 represents a comparison between the selected projects architectures, using the features previously explained. The form of reading the table is through to check if appears a dot or a dash, or with a description, in the case of appearing a dot represents the feature exists in that architecture, and a dash means the opposite.

Table 1: Reference Architecture Projects Analysis.

	SENSEI	IoT-A	OSMOSE
Interoperability	✓	✓	✓
Device Discovery and Management	×	×	×
Context-Awareness	✓	×	✓
Scalability	×	✓	✓
Management of Large Volumes of Data	×	×	✓
Security, Privacy, and Integrity	✓	✓	✓
Dynamic Adaptation	×	×	×

The Internet of Things Architecture (IoT-A) is a guideline to support the users in the design of their own IoT solution, addressing the structural concerns related to the IoT and proposing an Architectural Reference Model (ARM), starting with the definition of an initial set of key building blocks regarding functionality, scalability, performance, deployment and security, with the purpose to eventually derive into a large set of concrete IoT-Architectures [19]. This architecture is very complete and represents one step further in defining the IoT reference architecture, but it lacks the applicability that other more specific ontologies have proposed due to a more care for specific implementations and real world situations.

Another IoT model is the Semantic Sensor Network (SSN) Ontology[1] uses an OWL[2] (language designed to represent complex knowledge about objects, groups of objects and their relationships) ontology to describe sensors, their capabilities, measurement processes and resultant observations. This ontology contains only concepts and relations directly relevant to the sensors, leaving aside concepts related to other domains. In this way, the ontology is better positioned to provide modularity and reusability [20]. This OWL ontology, describes a specific straightforward approach to sensors, sensing, measurement capabilities of sensors, observations that result from sensing and deployments, or field applications, in which the sensors are used. The network is visualized in modules to provide better reusability and adaptation based on sensor observation. It lacks additional information regarding sensor relative entities and the domains where they are used, which means that this ontology has to be complemented with other ontologies or models in order to develop an efficient practical solution.

The IoT Lite is a lightweight ontology, based on the SSN Ontology, to represent IoT resources, entities and services [21]. By creating a representation, in a more lightweight approach than previously existed in other ontologies, it is possible to achieve an ontology able to provide shorter response time and thus create a more efficient structure for systems. IoT-Lite is meant to be used with a quantity taxonomy, which allows discovery and interoperability of IoT resources in heterogeneous platforms using a common vocabulary. The IoT Lite Ontology is a meta-ontology designed to be extended in order to represent IoT concepts. It also focuses more on sensing and establishes a high-level concept on actuation, which allows any future developments or adaptations on this area. It is a lighter view of the SSN Ontology, ideal for environments or specific situations in such environments, which require fast and easy processing [21].

Other project that developed an IoT model is the OSMOSE Project[3], it designed and developed a reference architecture for modeling and managing the referred sensing liquid enterprise integrating [22], a view of three interconnected worlds: the **Real World (RW)**; the **Digital World (DW)**; and the **Virtual World (VW)**. The worlds represent three types of data management environments [23]. OSMOSE follows the osmosis metaphor concept, which relates to the process of passing the molecules from a less concentrated solution (individual perception of each world) to a more concentrated one. Following the sensing liquid enterprise paradigm, osmosis processes are a special type of business processes used to moderate the information exchanged among the worlds. Since the notion of osmosis processes is conceptual, they can be modeled using the same strategies of regular business processes.

C. Complex Event Processing

Complex Event Processing (CEP) is an emerging network technology that creates actionable, situational knowledge from distributed message-based systems, databases and applications in real time or near real time. It can deliver the capability to define, manage and predict events, situations, exceptional conditions, opportunities and threats in complex networks. Implementation scenarios range from network management to business optimization, resulting in enhanced situational knowledge, increased business agility, and the ability to more accurately (and rapidly) sense, detect and respond to events and situations. The use of a highly scalable and dynamic solution such as CEP can contribute to extract higher level of knowledge from situational information abstracted from processing business-sensory information [24]. In this work, the events are used to monitor key performance indicators (KPIs) defined by the user, since each device has their own role to

[1] https://www.w3.org/2005/Incubator/ssn/ssnx/ssn

[2] https://www.w3.org/OWL/

[3] www.osmose-project.eu

play. The CEP with the support of services can warn the user and monitor the devices about specific occurrences, which the key performances indicators that the user believes are needed to be monitored. For this reason, appears the need of doing a study in CEP engines to decide the best one to use in this work: **EsperTech CEP**[4] – is an open source for runtime architecture, platform administration and event processing features. It is an engine embeddable in Java architecture, with strong CEP feature set and open source status make it a top candidate to be embedded in other vendor tools or applications [25]; **WSO2 CEP**[5] – is a lightweight, easy-to-use, open source CEP server for real-time analytics of events. Detects events in real-time, actuating on the most significant events that are in the event cloud, being an high performing and massively scalable, supported by features like event partitions, mapping database values to events [26]; and **Fitman Dynamic CEP**[6] – is a CEP that support complex real-time processing pipelines, resolving the issue of data-driven detection. It was created focusing the smart factories, and based on technology provided by Espertech, being extended to react more expediently to particular situations [27].

Table 2: CEP Analysis.

	Esper CEP	WSO2 CEP	FITMAN Dynamic CEP
Open Source	✓	✓	✓
Community Support	✓	×	×
Technical Support	×	✓	×
Easy Integration	✓	×	×
Dashboard	✓	✓	✓
Latency	10µs	17µs	+10µs
Scalability	10 to 200K	+100K	+10 to 200K
Recovery	✓	✓	×

In Table 2 a comparison between the different presented CEP is made. The comparison is made through eight different features, more can be added, yet these ones were selected considering their relevance for the CEP to be chosen for this work. The form of reading the table is equal to the description made into the Table 1. The first feature determines if the software is an open source or not, the second and third shown if the software has community support and technical support. The next feature is to identify if the software have easy integration. The Dashboard is a feature that facilitates the monitor of the integration of the software. The Latency is the time interval between the events, and the Scalability is the capacity of the CEP to handle a growing amount of events. The Recovery option is the capacity for the system to recovery from a forced shut down and restore to an earlier point in time.

III. IoT Monitoring and Device Management for Interoperability in a Physical Network

In order to increase efficiency and productivity, device management is a must have feature for a sensor network. To reach this step several solutions can be used, this work is focuses in interoperability and sustainability of the device's network as a solution for the problem. An effectively and intelligent management of the enormous heterogeneity of devices and the data exchange between them is a step forward to improve network efficiency. A rapid implementation of adding new devices to a network promotes a sustainable interoperable environment enhanced by the management of devices. A device can be identified according to its properties/capabilities or by using a unique identifier like, for instance, a serial number. For example, a barometer can be identified using a serial number (xxx-xxx), another way to identify this barometer is by its specific properties, such as, model, measurement unit, etc., and within a reasoning process the device could be identified. An interoperable device management framework can be implemented using both forms.

This work proposes a solution based on architecture for supporting the addition of new devices to a network, by automatically providing the necessary services and events, related with that specific device to be used during its life-cycle. This method is supported by a model (to represent devices in the virtual world), a knowledge base (to reuse information, and harmonization), web services (services used to access their data or to send information) and a CEP (used to manage the events pre-defined by the system middleware). An automatically pre-configuration of a new device is made by an administrator not only to supply a connection to the framework, but also, in providing services and events that are pre-define by the system middleware. The user is then free to decide if any other service/event needs to be added, providing a higher level of freedom in terms of configuration and setup. The system middleware responsible for the installation of a new device and orchestration is called DEVMAN. The DEVMAN framework provides an intelligent way to give the additional services and events necessary for an effective, interoperable and sustainable solution.

A. DEVMAN Methodology

Fig. 1 illustrates the methodology of the DEVMAN framework. It describes how DEVMAN reacts in the moment that a new device is detected in the network. The methodology is divided in three stages. Stage one is called virtualization that starts with a detection of a new device, by using an installation web-service, the DEVMAN identifies the type of device (e.g. a temperature, humidity sensor, etc.), supported by a device ontology. This ontology aims to standardize the concepts of the device types to avoid semantic interoperability issues. The need of identifying the type of device with the brand and model is going to be supported by a KB of devices. This step is possible because of the existence of several brands, each brand has several models, but the models have information that is always equal in that specific model, making it possible to reuse it. In stage two, runtime preparation, by using the information retrieved from the KB containing the device properties, it then instantiates a new device and associates its pertinent services and events. The final stage, monitoring, represents a state where the device is inserted inside the platform and provides information to the framework thought services that in turn may or may not trigger actions in response to events, according to the device installation parameters.

[4] http://www.espertech.com/

[5] http://wso2.com/products/complex-event-processor/

[6] http://catalogue.fitman.atosresearch.eu/enablers/dynamiccep

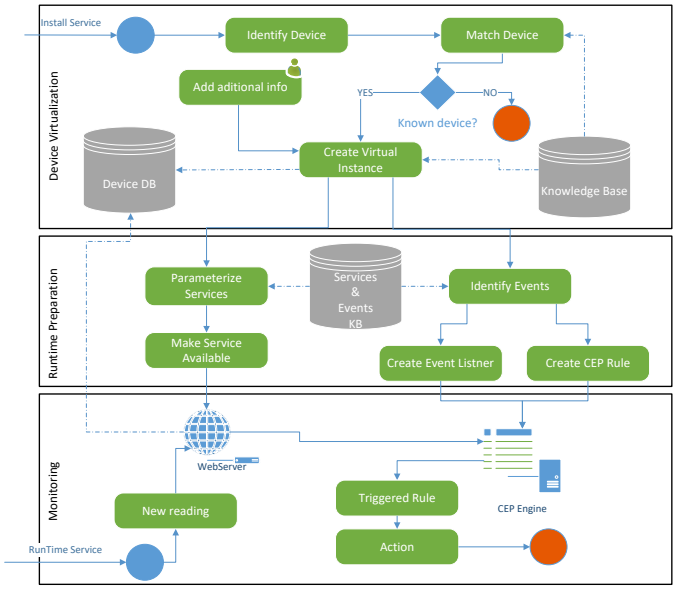

Fig. 1. DEVMAN Methodology.

To support stage one (Virtualization), in the detection of devices based on the KB, the methodology as completed with an example with a more detailed explanation about the matching device process, which is represented in Fig. 2. It is divided in three main steps or questions “The device exists in the KB?”, "Exists Type in KB" and "Reuse sibling”. Staring with the first question, in case of a positive answer, the device already exists, the middleware creates the virtual entity of the device (in this example, the sensor Sharp SharpIRxyt, which is an IR sensor) and its corresponding attributes. In case the device does not exist in the KB, the middleware is going to do one of following options. If it cannot find a device of the same type, the process will be terminated. On the other hand it will search in the KB for a similar type (in this example, there are two different similar devices), then ask the user if one of them can be used as a base device, in order to avoid a manual insertion of the relevant information. Of course it can happen that there are no similar models of the same brand in the KB, in this situation the methodology redirects the question to the user to choose or not a sibling. In case of positive answer it will create the virtual device instance with additional information, specific to that device, provided by the user. On the opposite side, it will request the user information to add a new device type to the KB. With this methodology implemented in the middleware, the insertion of information of a device is facilitated, since the intelligent middleware searches the KB in a tree-like basis where it will look for a leaf node, branching out to find a proper match.

The following phase provides support to stage two (Runtime preparation) of the DEVMAN Methodology. At this moment the virtual representation of the device is created in the database, the next phase of the methodology is ready to start. This phase is responsible for attaching services and events to the device, for this task the information provided by the KB is reused since the system middleware already has a similar device stored. In this automatic process, the KB provides the relevant information for events and services that the device is going to use during it is lifespan and to cooperate with the network. A list of generic services exists to make available to the devices, for each device like getting and setting data. Regarding events the procedure is different, according to the specifications and constraints provided by the KB a rule is instantiated inside a CEP, after which a listener is created that will trigger the necessary action (for example in the case of a temperature sensor, the KB provides a maximum temperature set-point that will trigger a warning). After this, the device is registered and ready to be used by the network. The final stage (Monitoring) is when the devices start to register the collected data using Runtime services, and the events trigger the specified actions according to their installation specifications.

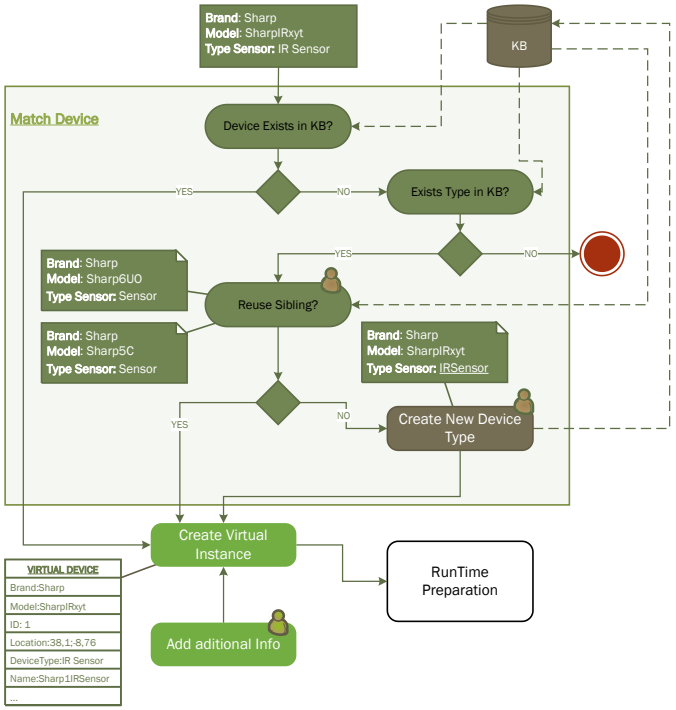

Fig. 2. Guide Example for Adding a New Device.

B. *Architecture for the Sustainable Interoperability of a Device Network*

In Fig. 3 the architecture is illustrated, it aims to manage the devices and connect them with the cloud platform. To reach this objective, the architecture is supported on a model to manage all the characteristics and features of the devices. Several components are used by the architecture to support the main objective such as, services, events, ontologies KB, etc. Using the project’s description from the previous sections, services and events are mainly based on the IoT-A project while the ontologies and knowledge base are derived from the OSMOSE project. As Fig. 3 shows, the architecture is divided into six components, which are described below:

1) Device Layer

Represents the physical part in the architecture (e.g. Arduino, Raspberry Pi, etc.) that sends the necessary information depending on the nature of the device to the cloud via the services component (web-services) using TCP/IP.

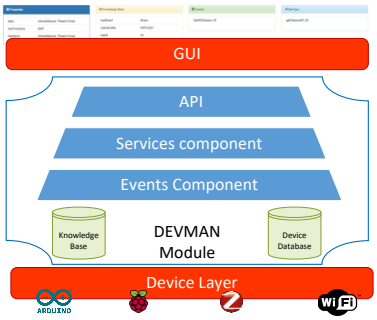

Fig. 3. Architecture for Adding New Devices to a Network.

2) DEVMAN Module

This core module, was implemented according to the DEVMAN methodology, and is responsible for identifying patterns during the device registration, and using these patterns to maximize the automation of registering other devices. This is made by proposing possible registration of each device, as defining their characteristics, proposing the services and events that can be used by them. This feature is supported by a knowledge base, which is continuously being enriched overtime with the experience of others similar devices and a database that provides storage for the initial setup requirements, in terms of, which services and events need to be associated to a specific device.

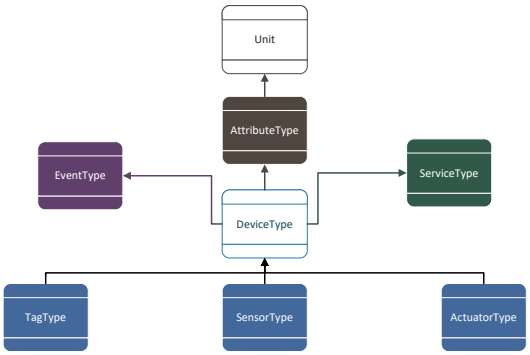

Fig. 4. Knowledge Base.

Knowledge Base – The knowledge base accomplishes two objectives, as a concept taxonomy (this taxonomy is used to harmonize the concepts used by the device database, as type of sensors, actuators, unit, etc.), and to have a real time taxonomy of the device characteristics to support the automatic insertion of a device. There many different devices, yet the same model can be used several times, therefore, this taxonomy stores this data and uses it when a similar device appears again, this will allow to do a semi-automatic insertion of the device, since it will have the same characteristics. During this work several models were study to select the appropriate model to be used in this module. It was selected the model described in the IoT-A project as basis for this work. Fig. 4 presents the knowledge base structure, resembling the model, where a device can be classified in the three different types to help separate the concept of each device. An ontology of services relates to device in order to present the list available services that a device should have associated with. An ontology of events is also used here to reveal what type of events should be triggered by device type. The Attribute type represents the non-general attributes that a device can have that usually represent the main feature of the device. The unit part, expresses the unit which describes that particular attribute.

Service Component – This component provides and exposes model information not only to provide access but also to give events a way to report contextual awareness alerts or notification to business stakeholders outside of the network. The Services component is divided in two parts. The InstallService (services used to registry a device in the model), is a mandatory pre-configuration, after this registration the device is "visible" to the rest of the network. At this point the device is registered in the model, defined the characteristics and depending of the features of the device is added the services and events that can be consumed by the device). The RunTimeService (services used after the registration of the device is completed) is used by each device during their time in the network. Since each device is different from the other, several services exist and can be used by them. In order to use a service description becomes important, since the service exposes capabilities, attributes or functionalities, either as output data type or as an input parameter.

Events Component – In a network, each device has a purpose, as to monitor or to send notifications to another device(s). The events are used to facilitate this purpose, this module is the engine where the rules of the events are created and run over the life time of device, triggering the necessary actions through the services module. Using the ontology defined by the OSMOSE project, an event can be categorized as: Message Event; Timer Event; or Error Event; etc., others classifications are possible. The discussion made about CEP engines was to support the selection the appropriate CEP to use in this module, which was chosen the EsperTech CEP.

API – This module is the set of routines to expose the functionalities of the other different modules including the GUI, through the use of the web-services.

3) Graphical User Interface (GUI)

This module provides a visual representation of the entities as well as interfaces relevant to properly manage the devices.

IV. Metalwork Manufacturing Scenario

This scenario is related with a metal working company, which produces iron tubes. The company produces three types of tubes (square, round and flat), which can have different measurements, for example the round tube can have 50, 100 or more millimeters (mm) of radius. Production is done according to demand, meaning that they produce depending on the orders they receive, implying that there is no stock of material or product. This situation forces the company to have a more rigorous control on their stock, since it needs to buy more raw materials every time an order is received. Due to these facts, the company wants to improve their production by having more control over it, to know in real time the consumption of the raw material, allowing the order of new raw materials to be done on time, avoiding delays in the production.

One way to improve the production is through the use of sensors to monitor the production, since it enables the

identification of the raw material used in the moment of production, the tube that is being produced, verify the quantity of the order and check if the status of the production is ready for delivery, being divided into five phases, see Fig. 5.

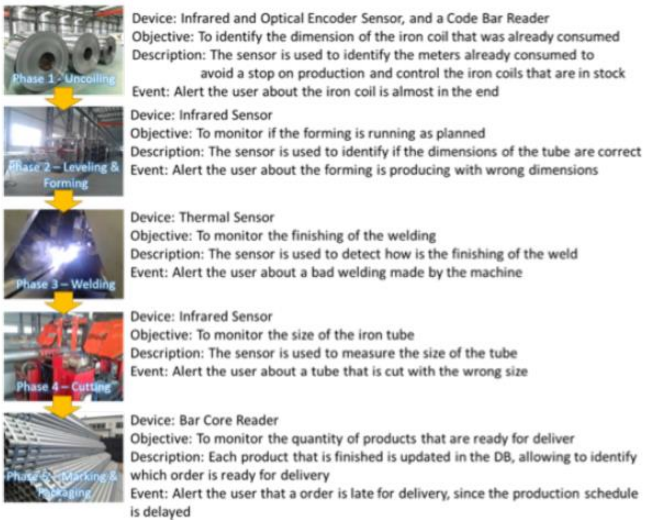

Fig. 5. Production Phases Description.

The first phase is the uncoiling of the iron coil, at this point the production starts with the preparation of the iron, enabling the ability to control the raw material being used and the standpoint of the iron coil, with the support of sensors. For the sensors to give an advantage in this phase, it was decided to use three different sensors, one Code Bar Reader to read the code attached to the iron coil to identify the coil in use. The Infra-red Sensor is used to identify the beginning and ending of the iron coil, measuring the distance between the sensor and the coil. Every time the coil is missing, the sensor is going to measure the distance to the floor, identifying that the coil is missing. The last one, is the Optical Encoder Sensor, which is attached to the belt of the production line, it measures the length consumed by the machine, allowing the system middleware to know (with the support of the Infra-red sensor) the length that is missing of the coil. With these three sensors working together, it is possible to have events to warn the user about the stock of the raw material (since sensors identify the coil that is being consumed). The production is stopped or not (due to the sensor that is identifying if coil exists in the production line), or about the missing length of the coil to complete the ordered quantity (to avoid a stop in the production to add a new coil).

The second phase is the leveling and forming of the iron, which is going to transform the iron plate into a tube. This phase passes through several stages until it reaches the final stage. Depending of the final form of the tube, the sensor is able to measure the dimensions of it and verify if the tube has the required dimensions. The system middleware needs to know which order is being executed to verify that all the settings are according to the requirements. For example, if it is producing a round iron tube with 50 mm of radius, the sensor is going to measure the radius to verify if it is correct or not. In the case of error the user can be warned through an event. The next phase is to do the welding of the iron tube, using the Infra-red sensor to control the weld. The sensor follows the weld to detect if mistakes occurs, warning the user through an event. The fourth phase relates with the production of the final product, which is to cut the tube with the respective length, usually these type of tubes have 6 meters of length. In order to identify if the correct tube length is correct or not, an Infra-red sensor is used, in case the measure is incorrect an event is triggered to warn the user. This event brings a big benefit, since it also allows to identify if the machine is not well calibrated avoiding the waste of more raw materials and time.

The final phase of this process is the packaging of the iron tubes, in this phase the machine packages the iron tubes, ready to be shipped for delivery. The use of a Code Bar Reader in this phase, benefits the tracking of the orders, therefore identifying if an order is fulfilled and ready for distribution. Every time that an order is not fulfilled or delayed, an event is triggered, warning the user. Since this occur in real-time, it is possible to take measures to avoid a more delays.

After being made a survey of the existing sensors, the sensors network is created in the cloud platform. This is made by creating the physical entity representation of each sensor (assuming the sensors already exist in the KB, the platform cloud is going to purpose the sensor to avoid the user to add it manually). Then, it is added to the sensor their characteristics (as type of sensor, name, protocols, etc.), the conceptual events and services that the platform cloud provides to this type of sensor (for example a temperature sensor by default, has a service to get the temperature and another to store values in the DB, and an event to detect a maximum or minimum temperature and warn the user). After the insertion of the physical entity, the framework detects a new entry and virtualizes the sensor. This is made by creating the virtual entity, and associates the specific events and services (based on the conceptual events and services provided by the KB).

After these steps a sensor is registered in the cloud platform, being able to provide the sensed data to the platform cloud, storing it to be later used for events or, for example, in a simulation scenario. This ends the installation phase, and starts the run time phase. In this phase, the monitor starts with the support of events to detect the performance indicators defined by the user and the consequential actions (warnings, alerts, etc.). A service is called to get the values of the sensor, occurring every time a new value is read. Then, an event is triggered due to the measured length, read by the sensor, indicates that the coil is missing, or finished. After which an event is triggered, warning the user, to change the iron coil. Only the process of one sensor is described with a practical example. All the steps are equal to the rest of the sensors, the only difference is which services and events are provided by the platform cloud depending on the type of the sensor.

V. Conclusions and Future Remarks

For the past few years, the IoT has been transforming businesses, increasing efficiency and productivity, by using different devices to provide information about the surrounding environment. IoT envisions to inter-connect and relate all these devices seamlessly, promoting cooperation with a high level of interoperability. Nowadays, industries use targeted solutions to address their specific issues, creating a problem of

interoperability when cooperation is required. To this date, many research projects address this vision of IoT, providing models, methodologies, interfaces and processes etc. However, each passing year, not only the number of devices increases but also new ones appear on the market with more functionalities and capabilities. Therefore, the question arises of how to inter-connect and relate all these devices in order to support fulfill the vision of IoT. The problem of having a heterogeneous world of IoT devices with different functionalities and capabilities creates difficulties in creating a full interoperable solution. The level of difficulty increases when cooperation is required, in order to extract knowledge from raw data, which is a key factor, not only in terms of business but also to increase efficiency of industries processes.

This work focuses on describing a framework, called DEVMAN, for enabling a rapid interoperable solution for adding new devices to a network. The DEVMAN methodology describes the key process of matching a device according to a KB that is accomplished by using a tree-like data analysis to accommodate the device being registered. Although this may require intervention by a human-user, the process is still much more efficient than doing a complete manual registration. The DEVMAN framework, according to the methodology, creates the necessary service association (from generic services) to access not only the mapping of the general properties but also the specific attributes that mainly characterize the device and corresponding associated values. It also aids in creating and associated the appropriate set of rules (from generic events), to a device, inside the CEP in order to implement the necessary actions or alerts pertinent to that event. The use of Services as a doorway, contributes as a source of the CEP engine for providing the necessary event processing actions, giving a more scalable solution, in order to help provide a contextual aware environment. The DEVMAN methodology and DEVMAN framework, described in this work are a step forward in contributing to the vision of IoT, in helping businesses and industries processes.

For future work can be improved the intelligence behind the matching of the device, to accommodate more branches and nodes in the data analysis tree in order to increase matching efficiency, by using (for example) neuronal net. With the adoption of reference solutions that can implement standard protocols with privacy and security, which allow real time data collection, the manufacturing paradigm will forever change. With these changes will arise huge challenges, since industries will demand more intelligent systems.

ACKNOWLEDGMENT

The research leading to these results has received funding from the EC HORIZON2020 Program under grant agreement n° C2NET 636909 (http://www.c2net-project.eu/) and nº VF-OS 723710 (http://vf-os.eu/).

REFERENCES

[1] A. Bassi, M. Bauer, M. Fiedler, T. Kramp, K. Rob, S. Lange, and S. Meissner,"Enabling Things to Talk,"*Designing IoT solutions with the IoT Architectural Reference Model*,Springer,Ed.Springer, 2013, pp 249–278.

[2] P. Daugherty, P. Banerjee, W. Negm, and A. E. Alter, "Driving Unconventional Growth through the Industrial Internet of Things." Acceture, 2015.

[3] Ecicsson, "Ericsson Mobility Report - On the Pulse of the Network Society." 2016.

[4] S. Haller, A. Serbanati, M. Bauer, and F. Carrez, "A Domain Model for the Internet of Things," in *2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing Section II*, 2013, pp. 411–417.

[5] K. Rose, S. Eldridge, and L. Chapin, "The Internet of Things (IoT): An Overview." 2015.

[6] D. Bandyopadhyay and J. Sen, "Internet of Things -Applications and Challenges in Technology and Standardization," *J. Wirel. Pers. Commun.*, vol. 58, no. 1, 2011.

[7] O. Vermesan and P. Friess, "Internet of Things Strategic Research and Innovation Agenda," *Internet Things – From Res. Innov. to Mark. Deploy.*, p. 143, 2014.

[8] R. Minerva, A. Biru, and D. Rotondi, "Towards a Definition of the Internet of Things (IoT)." IEEE, 2015.

[9] C. Agostinho, C. Marques-Lucena, M. Sesana, A. Felic, K. Fischer, C. Rubattino, and J. Sarraipa, "Osmosis Process Development for Innovative Product Design and Validation," in *Volume 2B: Advanced Manufacturing*, 2015.

[10] J. Ko, B.-B. Lee, K. Lee, S. G. Hong, N. Kim, and J. Paek, "Sensor Virtualization Module: Virtualizing IoT Devices on Mobile Smartphones for Effective Sensor Data Management," *Int. J. Distrib. Sens. Networks*, vol. 2015, pp. 1–10, 2015.

[11] J. Sahoo, S. Mohapatra, and R. Lath, "Virtualization: A Survey on Concepts, Taxonomy and Associated Security Issues," in *2010 Second International Conference on Computer and Network Technology*, 2010, pp. 222–226.

[12] T. Guo, T. G. Papaioannou, and K. Aberer, "Efficient Indexing and Query Processing of Model-View Sensor Data in the Cloud," *Big Data Res.*, vol. 1, pp. 52–65, Aug. 2014.

[13] S. Kumra, L. Sharma, Y. Khanna, and A. Chattri, "Analysing an Industrial Automation Pyramid and Providing Service Oriented Architecture," *Int. J. Eng. Trends Technol.*, vol. 3, no. 5, 2012.

[14] J. Shi, J. Wan, H. Yan, and H. Suo, "A survey of Cyber-Physical Systems," in *2011 International Conference on Wireless Communications and Signal Processing (WCSP)*, 2011, pp. 1–6.

[15] J. Lee, E. Lapira, S. Yang, and A. Kao, "Predictive Manufacturing System - Trends of Next-Generation Production Systems," *IFAC Proc. Vol.*, vol. 46, no. 7, pp. 150–156, May 2013.

[16] J. Lee, B. Bagheri, and H.-A. Kao, "A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems," *Manuf. Lett.*, vol. 3, pp. 18–23, Jan. 2015.

[17] Cisco, "The Internet of Things Reference Model," 2014.

[18] X. Li, M. Eckert, J.-F. Martinez, and G. Rubio, "Context Aware Middleware Architectures: Survey and Challenges," *Sensors*, vol. 15, no. 8, pp. 20570–20607, Aug. 2015.

[19] M. Unis, A. Nettsträter, F. Iml, J. Stefa, C. S. D. Suni, A. Salinas, and U. Sapienza, "Internet of Things – Architecture IoT - A Final architectural reference model for the IoT v3." 257521 IoT-A Project, 2013.

[20] M. Compton, P. Barnaghi, and L. Bermudez, "The SSN Ontology of the Semantic Sensor Networks Incubator Group," *Web Semant. Sci. Serv. Agents World Wide Web*, vol. 17, pp. 25–32, 2012.

[21] M. Bermudez-edo, T. Elsaleh, P. Barnaghi, and K. Taylor, "IoT-Lite Ontology," 2015. [Online]. Available: https://www.w3.org/Submission/iot-lite/.

[22] F. Lampathaki, H. P. Athens, P. Kokkinakos, C. Tucci, I. Alvertis, J. Psarras, S. Koussouris, and G. Viscusi, "FutureEnterprise, A Roadmap for Sensing Enterprise," 2015.

[23] C. Agostinho, M. Sesana, R. Jardim-Gonçalves, and S. Gusmeroli, "Model-driven Service Engineering Towards the Manufacturing Liquid-sensing Enterprise," *MODELSWARD2015 - Proceedings of the 3rd International Conference on Model-Driven Engineering and Software Development, ESEO, Angers, France, 9-11 February, 2015*, pp. 608–617.

[24] GemStone, "GemFire Real-Time Events: White Paper." p. 16, 2005.

[25] HTC Gllobal Services, "Managing Real-time Data Streaming Effectively with CEP Solutions - White Paper." p. 13, 2016.

[26] WSO2, "WSO2 Complex Event Processor." 2015.

[27] FITMAN, "FITMAN - DynamicCEP." 2015.