Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 50 (2016) 512 – 517

www.elsevier.com/locate/procedia

26th CIRP Design Conference

Integration of digital factory with smart factory based on Internet of Things

Navid Shariatzadeh[a,*], Thomas Lundholm[a], Lars Lindberg[a], Gunilla Sivard[a]

[a]Royal Institute of technology, Production Engineering, Brinellvägen 68, Stockholm 10044, Sweden

* Corresponding author. Tel.: +46-8-790-8338; fax: +46-8-790-8338. E-mail address: navid@kth.se

Abstract

Internet of things (IoT) in manufacturing can be defined as a future where every day physical objects in the shop floor, people and systems (things) are connected by the Internet to build services critical to the manufacturing. Smart factory is a way towards a factory-of-things, which is very much aligned with IoT. IoT not only deals with smart connections between physical objects but also with the interaction with different IT tools used within the digital factory. Data and information come from heterogeneous IT systems and from different domains, viewpoints, levels of granularity and life cycle phases causing potential inconsistencies in the data sharing, preventing interoperability. Hence, our aim is to investigate approaches and principles when integrating the digital factory, IT tools and IoT in manufacturing in a heterogeneous IT environment to ensure data consistency. In particular this paper suggests an approach to identify what, when and how information should be integrated. Secondly it suggests integration between IoT and PLM platforms using semantic web technologies and Open Services for Lifecycle Collaboration (OSLC) standard on tool interoperability.

Peer-review under responsibility of the organizing committee of the 26th CIRP Design Conference

Keywords: Digital factory; Internet of Things; Smart factory

1. Introduction

Internet of things (IoT) is defined as the interconnection via the Internet of computing devices embedded in everyday objects, enabling them to send and receive data [1]. In Smart factory products, resources and processes are characterized by cyber-physical systems (CPS) [2]. CPS is analogous to the Internet of Things (IoT) sharing the same architecture, however, CPS presents a higher combination and coordination between physical and computational components of production systems [3]. The digital factory is a model of a planned or real factory used for design, planning and operations .In the smart factory, the digital factory developed during engineering should be integrated with the smart factory with its real time data and inferred statistics and information. One significant capability is thus the integration of the digital factory with the smart factory. This capability includes the ability to create interfaces of digital things which are linked with physical things. Further, functionalities needs to be implemented for receiving data from the IT applications of the digital factory to the IoT platform which implements the smart factory, and providing feedback to the digital factory through IoT services. However, in the IT environment, data and physical resources are typically heterogeneous and a good integration strategy is needed to assure the consistency of the data which is pushed to or pulled from the IoT platform. There is a need for a common language for presentation and representation of data together with a protocol that enables IoT devices to communicate to the digital factory. For interoperability within a digital factory, many ontologies and information standards such as ISO 10303 have been developed. In the smart factory, the Resource Description Framework (RDF) is used to achieve interoperability [4], for instance, Semantic Sensor Network (SSN) [5] answers the need for a domain-independent and end-to-end model for sensing applications by merging sensor-focused, observation-focused and system-focused views.

The integration of digital factory and smart factory in a holistic way has not been considered in research. Furthermore, companies develop vendor specific solutions for their own IT architectures. They do not standardize the services and functionalities and they lack either semantic or

Peer-review under responsibility of the organizing committee of the 26th CIRP Design Conference
doi:10.1016/j.procir.2016.05.050

syntactic solutions. Hence, This paper suggests a solution for integration which not only encompasses standardized protocols and information models to exchange data, but also a methodology for the necessary steps which must be taken to adapt a general solution among the current approaches and technologies to achieve interoperability. It is based on an approach to integrate the IoT platform of the virtualized factory with the PLM platform of the digital factory. IoT platform is a type of cloud which can store real time data, retrieve data and enable users to connect, create, analyse and experience things. First, the paper presents this platform-based system architecture. Secondly it presents a generic framework for creating communication interfaces between the two domains. The generic solutions is based on basic read and write, update operations, to be extended into more advanced service interfaces provided by IoT platforms.

Open Services for Lifecycle Collaboration (OSLC) initiative provides a minimalistic set of standardized information models. Assuming a loosely-coupled distributed architecture of tools and services, OSLC adopts the Linked Data (LD) approach to ensure data consistency across the data resources. Hence, this paper adapts it for developing integration specification.

2. OSLC as a specification for smart and digital factory integration

OSLC is an industrial effort which develops standards that make it easy and practical for software lifecycle tools to share data with one another. OSLC standards apply the principles of linked data (LD) and REST protocol to provide an interoperable web standards-based environment [6]. In other word it is a framework that standardizes the data format of data to be exchanged, the protocol to communicate data and services to create, read, update and delete (CRUD) data.

The LD framework allows linking between data from heterogeneous systems and it is considered a flexible approach that provides support for integrating data through different tools [7]. OSLC is built on web specifications and uses RDF as a fundamental data model. RDF is a framework that represents the LD and it provides a generic graph-based data model for describing resources, including their relationships with other resources. LD consists of two technologies; Uniform Resource Identifiers (URIs) and the HyperText Transfer Protocol (HTTP). Although HTTP may be expensive for many IoT devices, it can be beneficial for the web and IoT interoperability since it is developed originally for the web. That is one of the reasons to select OSLC as a specification for IoT and PLM integration in our approach.

An OSLC adapter is the software that represents tool data in the form of an OSLC resource and makes these resources available to other tools. These resources are available through web services. OSLC defines the concept of ServiceProvider for each tool adapters to expose containers of resource that is hosted by a tool for integration.

3. Why OSLC?

There are three ways to configure physical things, services and end users in an IoT context (see figure 1). In the first case from the left, physical entity, software, and the service are running on the same physical entity (a manufacturing resource). This is a configuration in which we have a powerful physical entity which can support for example the HTTP protocol and services are deployed on the physical entity. In the second case from the left, the service of the user is running in the cloud. The API used between the service client and the service, however, is the same. In the third configuration the service is not running on the physical entity, but in the cloud. This can be the configuration for a constrained device that may not be able to expose a user interface across the network For example, due to energy consumption limitation [8]. The third scenario is the subject of this work, in other words we investigate how we virtualize a physical resource in an IoT platform, how we process this data and extract information, how we integrate this derived information with digital factory information management application.

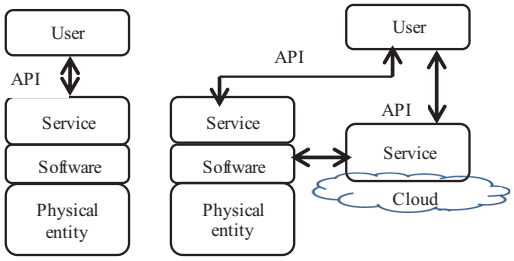

Fig. 1. Three types of configurations of physical entities and services

In order to provide structure to the Internet, network designers organize protocols. All protocols belong to one of five layers and provide services to the layer above. Figure 2 shows the traditional five-layer Internet protocol stack. The application layer is the top layer which is visible to the end-user; this is where the applications and their application-layer protocols reside [9].At this level the data is provided to the user.

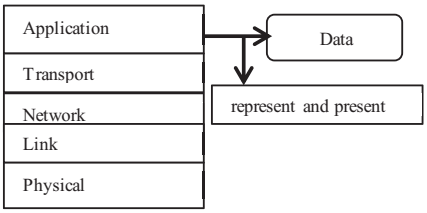

Fig. 2. Five layer internet protocol

CoAP is specialized web transfer protocol for use with constrained nodes and constrained networks in the IoT [10]. As mentioned, OSLC uses HTTP which can also be used as communication protocol of IoT devices if they are not constrained devices. This makes it a suitable framework for linking and integrating heterogeneous data. If IoT devices are constrained devices and use CoAP, OSLC still can be used since CoAP can be implemented of the REST pattern using HTTP. Hence, in this work we assume that physical entities work over internet protocol (IP) and that there are gateways

available that can be used in application layers for protocol translation in order to assist constrained devices to communicate with an IoT platform or other IoT devices.

4. System Architecture

Among IT applications in digital factory, PLM is critical for the integration and management of information. Hence we use a PLM system to represent the digital factory in this work. Figure 3 illustrates our system architecture for the integration between the 'IoT platform' and 'PLM'. The role of the PLM system is collecting and managing data of product definitions, processes, resources and decisions across the whole product lifecycle supporting changes and enabling traceability of these changes. Unique identifiers of products or parts are important for the PLM application during the lifecycle because products in PLM are used not only inside the enterprise but also in a distributed and collaborative environment. To fulfil the traceability and reusability of product/resource data, PLM systems usually follow a structured data model and usually use relational databases for persistent storage of data and vault (generally a protected file area) for storages of files [11]. They are always a client-server application, i.e., most of the application logic is executed on the server. The IoT layer is used for collecting large amounts of real time data, typically used for monitoring, controlling and planning on the shop floor. The real time information comes from various sources such as temperature pressure sensors or machine controllers. One difference between the IoT layer and the PLM layer is the database type to store data. In the IoT layer large amounts of data are collected and stored in an unstructured way and in real time databases. Real-time data bases in the IoT cloud are temporal, meaning that time is a dimension. For instance in IoT, individual stop times of a manufacturing resource are stored but after processing and converting data to statistics such as mean time between failures (MTBF), this can be stored in a PLM system for improvement of the resource. Temporal data are stored with timestamps and the data validity is lost after some time or is stored as historical data. For instance a resource temperature, as new temperatures come, the old temperature is not valid. NoSQL databases such as graph databases are more suitable for collection of real time data due to their ability for scaling up and capability of doing millions of data transactions per second.

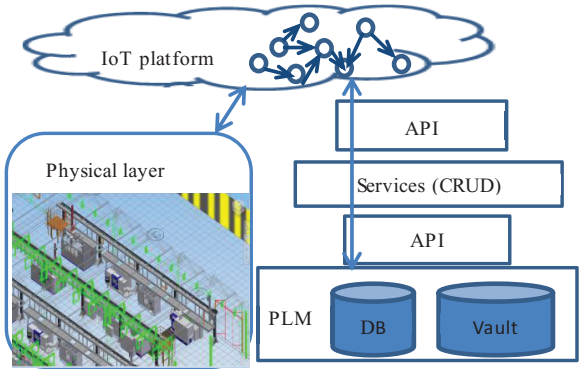

Fig. 3. System architect

Figure 4 demonstrates a summary of our idea to link data by creating OSLC resources for the same artifact in different systems and using OSLC core to create services to read, write, update or delete resources. This approach requires proprietary adaptors. The synchronization engine is responsible for the propagation and adjusting of required changes of artifacts in different systems when one artifact or one of its properties has been modified. Basically it is a time or event based procedure to harmonize the changes of integrated data in two platforms. For instance, when a new version of a file pushed to IoT is available in a PLM system, it should replace the former version or reversion. A common information model can be used to identify the integration structure in the IoT platform.

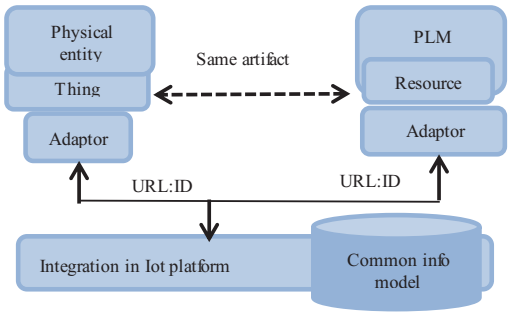

Fig. 4. OSLC Core concepts and relationships

The idea is that all devices interact with an IoT platform and the platform can communicate with other hubs such as a PLM system. Each hub has a catalog that allows the other hubs to access, search and update the data it can provide. A catalog is a specific type of resource representing a collection of resource items. Each item in a catalog refers to a single resource by its unique URI.

5. Suggested approach to integrate a digital and virtualized factory based on an IoT platform

5.1. Create domain model

The domain introduces the main concepts of the virtualized and digital factory. This step must be done by carefully analyzing individual domains to create sets of domain-specific concepts with properties and relations. Information from resources/products is heterogeneous and distributed. It is always necessary to build an abstract model to identify the overlapping concepts and their corresponding semantics among involved domains. Figure 5 demonstrates a domain model in a UML class diagram according to IoT_A proposed Architecture [12]. Physical articles are represented in the digital world by virtual entities. There are many kinds of virtual representations of physical entities such as machine tools or even temperature sensors. Virtual entities are linked to the single physical object that they represent. Each virtual entity must have one and only one ID that identifies it uniquely. Virtual entities are synchronized representations of a desired set of properties of the physical entity. This implies that the desired parameters representing the characteristics of the physical entity must be updated upon any change of the

physical entity attributes. For instance, if a temperature of a physical entity is changed, the temperature property of its corresponding virtual article must be updated as well. Digital entities are elements such as data-base entries, 3D models or other digital representations of the physical article in the PLM system. Devices are thus technical objects for connecting the real world of physical entities with the digital world of the Internet such as sensors and actuators. Services are functions that a virtual entity or physical entity can perform. A simple example of a service is a query written to a virtual thing to get the temperature of a particular component of a machine tool. Services can be orchestrated together in order to form a complete system. Virtual entity services provide access to information on a virtual entity level, process the collected data and trigger an action. An event is an action that occurs at an instance in time and changes a state in the system such as a value change in data exposed by a virtual entity. The event can be triggered within a service or trigger a service. Any virtual entity can subscribe to its own or another entity's events. When the event is fired, the source of the event passes the event data to the subscriber. For instance, when a temperature is higher than a predefined threshold a notification will be sent to a maintenance employee.

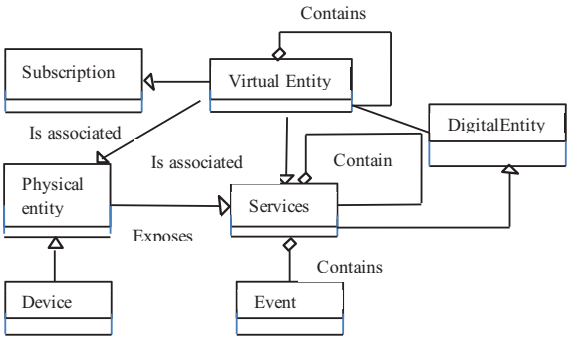

Fig. 5. Domain model

The following example represents an example of the domain modeling implemented in Thingworx which is an IoT platform. In order to monitor the changes that happen in an environment, two temperature sensors are installed in different locations. The aim is to announce to its users when it increase or decrease temperature due to changes it has noticed. To find the changes, the exponentially weighted moving average (EWMA) is used. EWMA creates averages of data in a way that gives less and gives less weight to older data [13].In this example; sensor services push temperature data every 20 seconds. We have two virtual things representing these two physical things in an IoT platform with one property (temperature). Then there is the "timer average thing" which is set to activate every 60 seconds. This thing has a service which gets the value from sensors A and B and creates the average of the temperature values. This requires a time interval to calculate the average as the input. The input is set to 60 seconds. In order to automatically use this service, a subscription is created which calls this service every 60 seconds. The output is then being saved into a data table which has 2 fields: A timestamp (unique ID) and the average Value field. Every 60 seconds, the subscription is adding a new row into the data table. Another service is created to convert the data from the data table into a JSON file. This can be called from outside with a simple REST call, and then it can be visualized in the browser of the JSON file.

5.2. Develop information model

The IoT information model defines the structure (e.g. relations, attributes) of all the information that is identified in the domain modeling step. This includes modeling of the overlapping concepts in the information flow, the data storage in the IoT platform and how these are associated. The information model details the modeling of virtual entities. It identifies attributes with a name and a type and one default value to which meta-information can be associated. As mentioned in section 3, important meta-information with respect to real time data are e.g. at what time a value was measured (i.e. timestamp) and the location where a measurement took place. Moreover, in this step the relationship between virtual entity and service is detailed in the sense that it relates a certain attribute of the virtual entity to a service. The IoT information model models all the concepts of the domain model that are to be explicitly represented and manipulated in the virtualized world. In addition, the IoT information model explicitly models relations between these concepts. The main domain model concepts that are explicitly represented in an IoT system are the virtual entity and digital entities and their corresponding services. As the virtual entity is the model of the physical entity in the digital world, there is no other representation of the physical entity as part of the IoT information model. The minimalistic set of concepts and their corresponding properties are selected according to the domain model to be represented in the IoT platform. Virtual entity attributes are properties of a physical entity that are required to be captured in the virtual world. The entity type may refer to concepts in the domain ontology in the previous step that may define what attributes a virtual entity of this type may have.

5.3. Communication model and protocol

Communication capabilities are relative to the type of data exchanged with a device (identifier + data, timestamps). Interoperability is achieved by using URIs as unique identifiers for things. The thing-URL has to be available over the network. If a device associated to a thing is not able to handle the URL of the thing, it is associated with a gateway which has to translate the RDF URL to its respective device specific identifier. Thanks to OSLC, the description of the representation of the information can be in XML, RDF+XML or JSON. JSON is more compact than XML and consequently it needs less space for data storage. It is important to clarify that virtual entities in IoT can also interact with non-IoT things. For example, a virtual entity could need certain information provided by an autonomous web application, a non-IoT entity, in order to make decisions. The ability of RDF to represent properties which are not part of the OSLC specification makes it possible to link any virtual entity

property to other resources on the web. When using RDF for information annotation URLs have to be accommodated in the information descriptions.

5.4 Identify type of data to be integrated between IoT and PLM

There are different types of data which need to be dealt with in IoT applications. The first type includes real time data representing the current status of the system. In IoT only the data read directly from a sensor can be considered as real time data. This type of data is not the typical target of a PLM system. The second type is derived data that have been created by analysis of raw data, for instance, the average energy consumption of a manufacturing resource in a specific time span. The third type of data is inferred data which is knowledge that has been inferred by applying logic or facts. For instance observing a specific pattern of vibration can be a sign of a machine failure. The second and third types of data are worthwhile to be part of a configuration management in a PLM system to be considered for product design improvement. A manufacturing resource, its structure and technical data are pieces of information which usually exist in a PLM system. This information may have to be integrated in an IoT platform to avoid manual instantiation. Using OSLC and linked data helps minimizing the amount of data to be copied between two systems. Another type of integration is file based. File integrations mean that the application files are managed by the PLM system, through some checkin-checkout mechanism. Here the PLM system knows nothing about the file content; it is just managed as bulk data. However, some metadata are generally managed, like file name and format, creation and modification dates, version and states. For instance, the last revision of the STEP NC file including product description, dimensions and tolerances, features, toolpath and operations, from a PLM system can be linked to an IoT platform, for an optimization purpose.

5.4. Information flow and permissions

After identifying the data which is transferred between the PLM system and IoT platform, the information flow mechanism must be identified. OSLC and linked data can support push pattern and request/response. The push pattern is a one-way communication between a PLM and IoT in which an IoT server sends data to a PLM server that receives the data. The IoT server in this case knows the addresses (URIs) in advance of the PLM OSLC services to create or update data. The request/response pattern is a synchronous or asynchronous way of communication between IoT server and PLM service resources. A server sends a request to a PLM OSLC server. The IoT server will receive the request and will send a response back to the PLM OSLC server. The IoT server is waiting for the response until the PLM OSLC server has sent it. There are two sets of permissions, one for design time and one for run time. The design time permissions are for managing who has the privilege of creating and modifying virtual entities. The run time permissions determine who can access data, execute services, and trigger events on a virtual entity. Users or a group of users can be embedded in one permission to be able to create, read, write and update data or refuse them.

5.5. Identify timing in integration

Timing is an important aspect of integrations. In general when a data item changes, the recent value must be reflected in the other platform using that data. It could also be available on request. However statistical data can be integrated in time intervals, for instance MTBF of a manufacturing resource is typically updated every three or six months. Another type of integration is on demand. For instance, when a user asks for the last measurement of a sensor, the value should be sent. Timing of integration should be identified by considering the information life cycle in the IoT platform. Some information might be stored permanently or have an expiry date after which the information is to be removed. It is also possible to store data for a specific time span and after that period only a portion of the data is kept while the rest is dispensed.

6. Case study

Our aim is to illustrate the proposed PLM and IoT platform integration concepts in a case study. The idea is to integrate data and IT tools in a way that various applications (services) can act on the data in parallel, enabling faster feedback between activities in PLM and IoT. The IT environment in this case is heterogeneous, based on the ARAS Innovator PLM system and Thingworx IoT platform. Here the domain of interest is monitoring of machine status with parameters such as temperature and pressure. From analysis of this domain the ontologies of domain-specific concepts with properties and relations are created, as well as an identification of what information is required from other domains. Consequently a set of minimalistic set of concepts with properties and relations to be shared has been identified. An adaptor has been developed to expose machine technical data stored in Innovator and integrate them with Thingworx. To get the real time data, a machine was simulated using Thingworx Java SDK which establishes and manages a secure AlwaysOn connection with the Thingworx server. The data are simulated by random numbers and pushed to the platform in specific time spans. Figure 6 illustrates the main flow of implementation. Manufacturing resources and their technical information are exposed using the OSLC core and are registered in the Thingworx platform in a table which basically is a JSON implementation. Secondly, a virtual entity is simulated. Hence real time data are pushed to the virtual entity. Then this real time data are bound to the original entity in the platform. Different services and events are also developed for monitoring the real time data. For instance if a temperature is beyond a specific threshold, then a user is notified using the defined subscription. Moreover, important events such “DataChange” are defined to update the changed properties. Derived information from Thingworx is exposed through services as JSON and XML and then with the help of Innovator API is integrated in Innovator. The case study implies that the suggested IT system architecture together

with OSLC can be used and implemented using the available technologies for integration purpose. The evaluation of the suggested approach must be also analysed with resource-constrained internet devices in future work of this research.

7. Conclusion

To achieve interoperability between a smart factory (real time data) and a digital factory three layers must be considered. First data transfer protocols, second, data representation and presentation and third semantics and understanding of data. There are many approaches, technologies and data models to accomplish interoperability on each level. However, in order to select the most suitable ones according to the desired goals, there is a need for guidelines to show the different steps that need to be taken to identify the best solution for integration. Hence this paper contributes to IoT domain integration with digital factory by developing an approach to identify what, when and how information should be integrated. Secondly it suggests integration between IoT and PLM platforms using semantic web technology and OSLC. OSLC uses HTTP as the data transport protocol and RDF and JSON as the data format. Another advantage is that it is an open standard; therefore everyone can contribute to it. Its openness also allows that any particular resource can be linked to other information in other hubs. It also couples comprehensive ontologies with light weight data formats like JSON which needs less space for storage and processor. Security in the IoT is a significant issue out of scope of this paper. However, it must be a critical aspect in the solutions for interoperability. The security issue will be the investigated in the future research of this work.

References

[1]http://www.oxforddictionaries.com/definition/American_english/internet_of_things.Accessed:2015-04-18

[2] Hermann. M, Pentek. T, Otto. B. Design Principles for Industry 4.0 Scenarios: A Literature Review; 2015.

[3] Radu. C, Ioana. A, Olteanu, Gheorghe. Smart Monitoring of Potato Crop: A Cyber-Physical System Architecture Model in the Field of Precision Agriculture. Conference Agriculture for Life for Agriculture; 2015.Vol 6 p. 73–79.

[4] Ryman A. Linked Data, RDF, and OSLC Resource Shapes: Define REST API Contracts for RDF Resource Representations; 2015.

[5] Hasemann. H, Kleine. O, Kroller, A, Leggieri. M, and Pfisterer. D. Annotating real-world objects using semantic entities in Wireless Sensor Networks. Springer; 2013. p. 67-82.

[4] Shariatzadeh. N, Sivard. G and Lindberg. G. An approach for manufacturing process representation in product lifecycle management. Key Engineering Materials ;2014.Vol. 572 p. 239-244..

[5] Open-services.net. OSLC Primer - Open Services for Lifecycle Collaboration. http://open-services.net/resources/tutorials/oslc-primer/; 2015.

[6] Lee B. Linked data-design issue. Available at www3.org/designIssues/ /linkeddata.html; 2006.

[7] OSLC Core Specification Workgroup. OSLC core specification version 2.0. Technical report, Open Services for Lifecycle Collaboration; 2010.

[8] Bauer. M. et al. Final architectural reference model for the IoT.http://www.iot-a.eu/arm . 2014.

[9] Braden. R (editor.). Requirements for Internet Hosts – Communication Layers. October 1989. https://tools.ietf.org/html/rfc1122

[10] Ishag. I, Carels. D, Teklemariam, K, Hoebeke. J, Abeele. F.IETF Standardization in the field of Internet of Things (IoT): A survey. Journal of Sensor and Actuator Network; 2013. P. 235-287.

[11] Shariatzadeh. N, Lindberg. L, Sivard. G. Rapid production changes through the coordination of factory layout models and activities. Journal of Applied Mechanical Engineering, 2014.

[12] Magerkurth. K (editor: SAP).The Internet of Things – Architecture, Deliverable D1.4 Converged architectural reference model for IoT. http://www.iot-a.eu/public/public-documents;2012.

[13] Patel. A, Divecha. J. Modified exponentially weighted moving average (EWMA) control chart for analytical process data. Journal of Chemical Engineering and materials Science;2011. Vol.2(1). P-12-20.

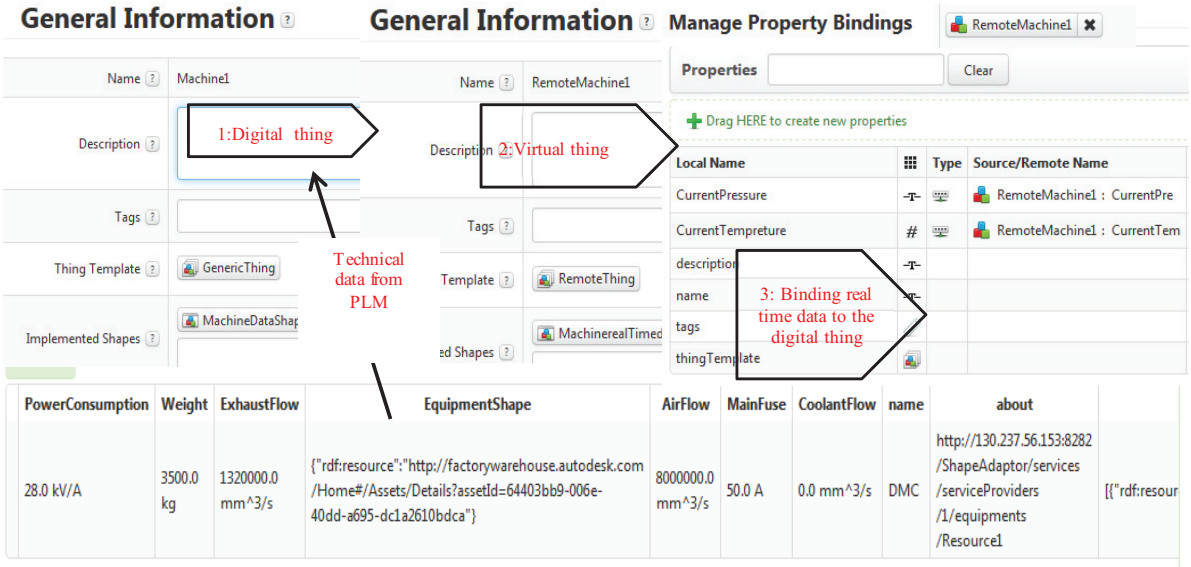

Fig. 6. Virtualization of a physical entity in an IoT platform and integration with its technical data from PLM system