DOI: 10.22213/2658-3658-2019-13-22

A Unified Ontology Namespace for Enterprise Integration – a Digital Twin Case Study

Joe David [1], Antti Martikkala [1], Andrei Lobov [1, 2], Minna Lanz [1]

[1] Automation Technology and Mechanical Engineering Unit, Tampere University, Tampere, Finland
E-mail: {joe.david, andrei.lobov, minna.lanz}@tuni.fi
[2] Mechanical and Industrial Engineering, Norwegian University of Science and Technology, Trondheim, Norway
E-mail: andrei.lobov@ntnu.no

Received: July 25, 2019

Abstract. In order to be able to scale business enterprises and digitally transform them, we need to be able to readily integrate business data. Traditional systems have business data travelling through different layers of the automation stack vertically. This study presents an architecture that employs a unified ontology namespace for integration of business data. Such an architecture is thought to enhance transparency and visibility easing integrations. A use-case exploiting such an architecture with webservices is presented to convey the ease at which different applications, such as a digital twin, can be "plugged in" to realize further business potential of enterprises.

Keywords: ontology, ERP, MES, PLM, digital twins, enterprise, web services, REST, SOAP, SPARQL

INTRODUCTION

Product Lifecycle Management (PLM), Enterprise Resource Planning (ERP) and Manufacturing Execution Systems (MES) have been the cornerstones of many (manufacturing) industries for decades but they have traditionally functioned as distinct and disparate systems coupled by means of proprietary protocols that prevent data from freely flowing across them. In order to obtain the best product in each level of the stack, often enterprises make use of different vendors at each level that have been implemented in completely different platforms and their convergence would generally take place only by means of software API integration done by software experts.

Fig. 1 shows the automation pyramid based on the standard that depicts the various level of automation in a factory. It constitutes of the following levels:

– **Level 0** is the field level contains the field devices, i.e. sensors that senses an event or measures one or more input variables and actuators that manipulate variables.

– **Level 1** is the control level that consists of programmable logic controllers (PLCs) or distributed control systems (DCS). They take in the information from the field layer to act upon them, make decisions based on programmed logic and pass back the outputs to the field layer.

– **Level 2** is the supervisory level that primarily uses supervisory control and data acquisition (SCADA) systems and other human machine interfaces (HMIs). Operators make use of these systems to monitor and control process data which may be stored in databases.

– **Level 3** is the planning level that constitutes mainly of the manufacturing execution systems (MES) that monitors the manufacturing processes in the plant that transforms products from its raw form to finished goods.

– **Level 4** is the business planning and logistics level that comprises mainly of the enterprise resource planning (ERP) systems that is a suite of computer applications to monitor business processes such as purchasing, sales, finance and payroll among others.

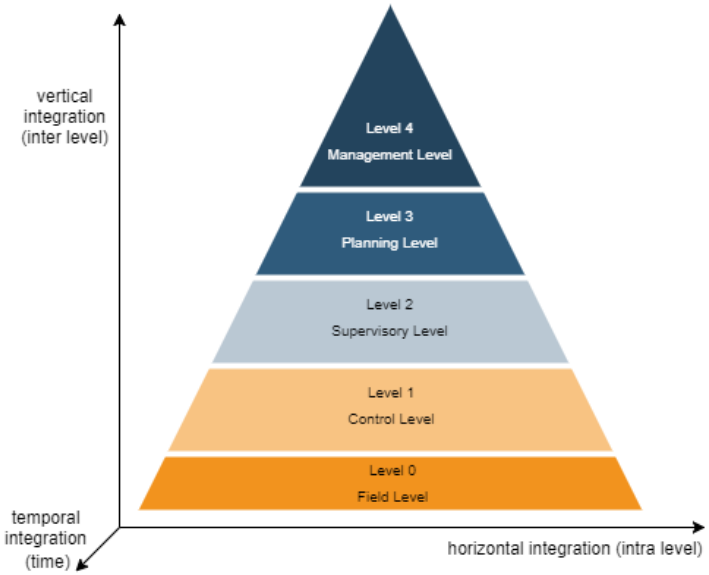

Figure 1. The Automation Pyramid

Integration in automation can take place along three directions, namely horizontal, vertical and temporal (longitudinal) (Sauter, 2005). Horizontal integration refers to the integration that takes place within each level (intra-level), vertical integration takes place along different levels (inter-level) and temporal integration takes place along the lifecycle of the plant.

To attain the holy grail of tri-system integration (PLM, ERP, MES), this paper envisions an architecture that publishes information from each of the layers of the pyramid to common unified knowledge base. Although the architecture allows for horizontal and vertical integration, the focus of the study is on temporal integration. Temporal integration refers to the changes that the automation system undergoes during its life-cycle. The different phases of the life-cycle include design, engineering, commissioning, operation and maintenance. Specifically, we look into reconfiguration of the automation system during operation as a use case. The unified architecture is expected to facilitate data sharing between constituent systems to offer visibility that would help eliminate redundant processes and waste, reduce product delivery times and ensure optimal overall running of the plant.

The remainder of the paper is structured as follows. The next section presents a background on the technologies underpinning the approach undertaken in the study and related work. The approach presents the conceptual architecture. A use-case where a digital twin for PLM is developed for a system that is developed based on the approach is discussed in the case study section. Discussions take place in the subsequent section and are followed by a conclusion where future work is presented.

BACKGROUND AND RELATED WORK

Work undertaken in this study revolves around two main technologies, web services that expose functionalities of applications as services and ontologies by means of which the domain knowledge or the current state of the system is represented. In this paper, the term web services is used loosely and does not necessarily mean connected to web, although this may be the case. It is used synonymous to Application Programming Interface (API) and is used to connote a medium of communication. Clarifying further, applications can make use of the service-oriented architecture using HypeTtext Transfer Protocol (HTTP) in local area networks without be connected to the World Wide Web.

Technologies

Webservices

A Web service is "software system designed to support interoperable machine-to-machine interaction over a network"("Web Services Glossary § Web service, W3C," 2004).Such a generic definition allows for REST architectures to be seen as one and consequently we look into two dominant paradigms in service oriented architectures, REST and the simple object access protocol (SOAP).

SOAP interactions occur with data in the XML format. SOAP web service descriptions are expressed using WSDL, an XML-based standard used to publish service descriptions using ports (service address), port types (operations that can be performed) and bindings (protocol used) (Gudgin et al., 2007). Universal Description, Discovery, and Integration (UDDI) is a standard for describing, publishing and discovering these services.

REST is the abbreviation for representational state transfer. A REST(ful) service is one that is built on a REST architecture that stipulates six architectural constraints (Fielding & Taylor, 2002): Uniform Interface, Stateless, Cacheable, Client-Server, Layered and Code-on-Demand. REST makes use of create, read update and delete (CRUD) operations using HTTP verbs typically on HTTP protocol. Web Application Description Language (WADL) (Haldey, 2009) is typically used to describe RESTful services .

Thus, while SOAP is a protocol that is formally defined using official web standards developed by the World Wide Web Consortium (W3C), a REST service is an architectural style that needs to be followed in order to qualify as a REST service. SOAP uses XML while REST can use a variety of formats including JSON, HTML, XML, etc. As such, SOAP requires more resources and bandwidth, while REST is rather lightweight and requires fewer resources. While SOAP services are mainly function driven, i.e. used for the transfer of structured information, REST services are mainly data driven and used for accessing a resource for data. As far as security of exchanged information is concerned, REST inherits the security features from the underlying transport protocol (HTTP(S)), while SOAP benefits from its own message level security via the WS-Security standard. Thus, REST offers only a faster point-to-point security while the data is being exchanged over the wire while SOAP offers a reliable end-to-end security.

When choosing between the two for a business application the following may be considered:

- **Complexity:** REST is to be used for simpler and faster access to a resource. SOAP's application is different in the sense that it is used mostly when maintaining open stateful connections with a complex client is necessary. REST transactions are stateless and independent of each other.

- **Data formats:** REST services can make use of data in various formats including JSON, CSV, XML, etc. while SOAP is limited to XML. XML is verbose and is less easy to parse than JSON or CSV and can accumulate on computation costs.
- **Standardization and Security:** If standardization is key, SOAP offers support for Web Services specifications. As for security, both support Secure Sockets Layer for point to point protection but SOAP offers WS-Security for end-to-end enterprise level protection.\\
- **Legacy applications:** Legacy systems are another argument in favour of SOAP. REST have gained popularity in recent times and many applications may still have only an implemented SOAP API.

Thus, it can be said that both REST and SOAP make use of different semantics and format to provide similar functionality and operate with separate security concerns.

Ontologies

Ontologies originated in research areas of metaphysics where philosophers used to describe the existence and nature of reality. A simple definition of ontology is that of Agarwal (2005) who states that "an ontology is, therefore, the manifestation of a shared understanding of a domain that is agreed between a number of agents and such agreement facilitates accurate and effective communications of meaning, which in turn leads to other benefits such as interoperability, reuse and sharing".

The rise of the semantic web has seen the emergence of ontology markup languages based on XML, of which Web Ontology Language (OWL), a W3C recommendation ("OWL - Semantic Web Standards"), is the most prominent. Ontologies have since been pursued by many, in various fields and domains to simplify and represent complex, unstructured and heterogeneous information as will be seen in the following section.

Related Work

Existing literature referring to ontology as a representation of manufacturing domain knowledge is plentiful. Specifically for enterprise integration that includes work of Zoubeidi, Kazar, Mesbahi, & Benharzallah, (2014) which presents an architecture towards integration of Enterprise Resource Planning systems (ERP) with a focus on semantic integration based on the context of use. The PABADIS'PROMISE (Diep, Alexakos, & Wagner, 2007) project deals with heterogeneity in a manufacturing environment and presents an interoperability framework based on ontology. Kalogeras, Gialelis, Alexakos, Georgoudakis, & Koubias (2006) present an architecture for vertical integration of the enterprise by using workflows to represent enterprise processes.

Different approaches to formalizing ontologies can be found as well. Nach & Lejeune, (2008) present an ontology that supports the ERPs for SMEs. A generic ontology for resources in an enterprise formulated from competency questions have has been presented by (Fadel, Fox, & Gruninger, 2002). MASON, a manufacturing upper ontology has been presented by Lemaignan, Siadat, Dantan, & Semenenko (2006).

APPROACH

The automation pyramid presented in the previous section saw a tightly coupled stack of applications that only interacted with applications in the adjacent layers. If for some reason we have to diverge from the traditional flow of information, say for example that we need some information from the ERP system to alert the operators at the SCADA layer, this would

have to flow through the MES system. This would require integrations at both the ERP-MES level and the MES-SCADA level. It is clear to see how the existing architecture is suboptimal. If we consider another scenario, where one would like to deploy an application that would use data from different layers of the stack, this would mean making discrete connections from distinct applications on disparate platforms, which is both time consuming and requires expert intervention.

Topology

Fig. 2 shows the conceptual architecture of the approach taken in this study. It consists of the systems that constitute the automation pyramid in Fig. 1. However, it takes the form of a star topology as opposed to the vertical stack. The central component is the domain ontology that each of the systems update to maintain a comprehensive updated knowledge model of the plant.

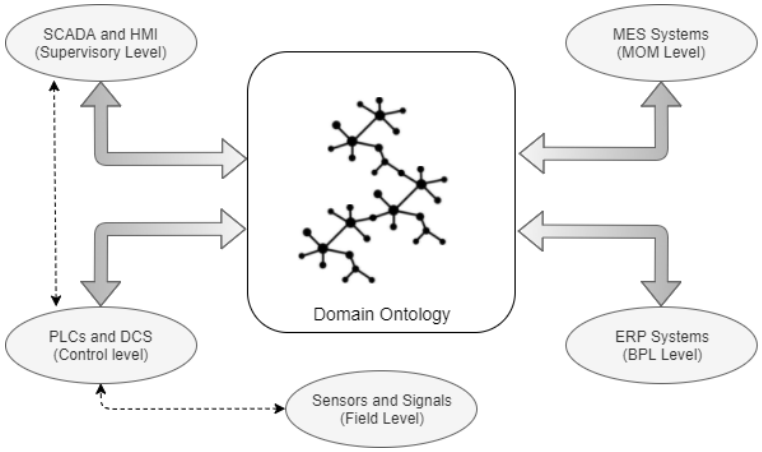

Figure 2. Conceptual Architecture

Discrete connections may still exist between these systems. For example, time critical applications may necessitate discrete connections from the field sensors and the control layer for real-time decision making based on programmed logic. This could be via any industrial (fieldbus) protocols, Profibus for example.

In some cases, especially in the case of legacy systems, the domain ontology acts as a knowledge base in addition to the existing vertical stack and does not replace it. This can be thought of as a generalization of the first point.

Information Interchange

Interaction of heterogeneous systems on disparate platforms requires standardized interfaces that foster information exchange between them and the unified knowledge base. Web services for long have provided just this interoperability is an obvious choice in the undertaken approach. There are mainly two kind of web services; web services that employs the simple object access protocol (SOAP) and RESTful web services, the relevant details of which was discussed in the previous section. We make use of both depending on the application requirements.

SOAP is to be used for applications where security is of prime importance or in cases where the endpoint under consideration is already implemented in SOAP. These include applications at the enterprise level such as payment gateways in financial services, customer re-

lationship management (CRM) systems, etc. REST is to be used for applications where efficiency involving a lightweight architecture is needed or where the endpoint of the application is a RESTful one. This would be in the case of field devices, or devices in the control (PLCs) and supervisory layer (SCADA Systems).

Knowledge Representation

Domain knowledge is represented using ontologies introduced in the earlier section. Fig. 3 shows the relationships between classes (entities) in the ontology. They are associated by means of need-feed relationships that exist between them. Here the relationship need takes the meaning of requires while feed relationships can be viewed as supply with or produces.

The Resource entity: A resource is any object that plays a role with respect to an activity (Fadel et al., 2002). As such, resources could be equipments such as work centers that process products, humans that perform activities that require human intervention such as loading of pallets, maintenance, electricity required to operate equipments, etc.

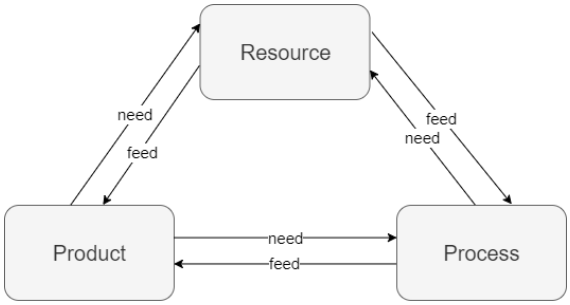

Figure 3. Relationship between entities constituting the ontology

The product entity: A product is the produce of the production system or line obtained via the transformation of raw goods.

The process entity: The transformation of raw goods to products takes place by the means of processes. The ontology developed is available online[1]. It must be noted that object and data property axioms are included for illustrtation purposes only. Those axioms are automatically created during runtime depending on the response from the production systems during runtime.

USE CASE – DIGITAL TWIN FOR PLM

This section presents an example of a use-case that utilizes the architecture described in the section III. The use-case is that of a digital twin of a production system that interacts with the domain knowledge to facilitate product lifecycle management functions.

Architecture

The conceptual architecture of the use-case is depicted in Fig. 4. It consists of a production system interfaced with a RESTful interface and its description which is stored in a WADL document that is made available for necessary integrations. A knowledge broker exists which acts as an intermediary between the source of the information in the production system and its ontological representation as the domain knowledge and consists of a web service discovery component and an ontology broker. It performs two basic functions: (1) Translation and transformation of necessary knowledge to be stored in the unified ontology namespace and (2) Discovery of web services to be able to perform the previous function. The knowledge base is developed in OWL remains synchronized with the current state of the production system with the help of the knowledge broker. The digital twin utilizes a SPARQL wrapper to query the knowledge base.

[1] https://github.com/joedavid91/unifiednamespace

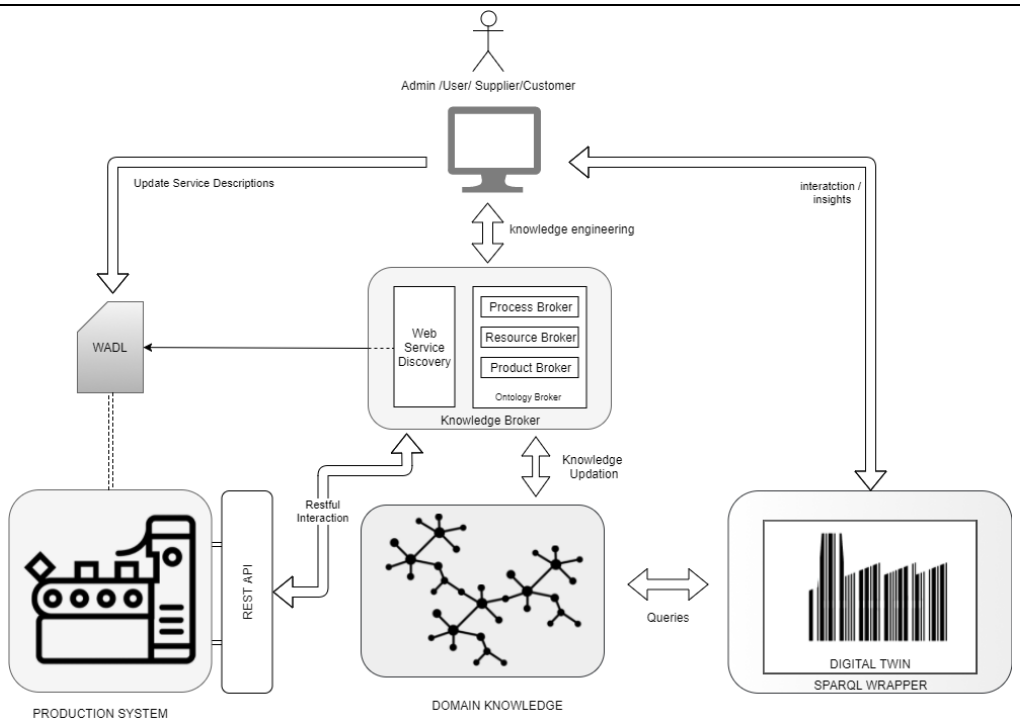

Figure 4. Realized Digital Twin Architecture

Implementation

The production line is that of a Flexible Manufacturing System (FMS), the specifics of which are introduced in an earlier work (David, Lobov, & Lanz, 2018). The system consists of components typical of an FM system such as machining centers, loading and material stations, washing machines, crane and its associated storage. The knowledge broker is a software component developed in python using libraries *urlib2* for the restful requests, *rdflib* to express the information in RDF, *wadllib* for the discovery of services. The knowledge base is developed using Protégé, an open source ontology editor and expressed in the Resource Description Framework. The domain knowledge can either exist as a file or can be served via HTTP via a Jena Fuseki Server implementing a SPARQL endpoint. The digital twin is developed using the software Visual Components, a leading developer of 3D simulation software for manufacturing. One of key concepts underlying the development of the digital twin is the separation of concerns by developing modular components. Each of the components exist independently and have a description of itself that is made available by its manufacturer in the XML format. The description contains all specifications of the component and describes it in its entirety. For example, a machining centre has information such as the PLC parameters, Pallet Changer parameters, Address parameters, Tool parameters, NC program parameters among a lot of others. The digital twin uses these specs from the domain knowledge to initialize the components in its 3D world. The digital twin behaviors are also implemented in python in Visual Components and makes use of the *SPARQLWrapper* python library to be able to query the domain knowledge. Thus, although the knowledge is centralized the intelligence is distributed.

Case Study

As a particular scenario, we examine a case that allows for automatic re-configuration of production systems during commissioning and operation and demonstrate how the architec-

ture envisioned in the earlier section allows for it. It must be noted that, the intention is not to deal with spontaneous dynamic networks at the protocol level but show that the digital twin can make use of such an architecture to incorporate dynamic networks.

The devices registered in the production line are made available in a device discovery API exposed through the REST interface. This service is discovered by the WS discovery component of the knowledge broker and the domain knowledge is updated accordingly. Fig. 5*a* shows the REST request and response from a third-party REST client. Similarly, the location coordinates of the devices are made available in a device data API exposed through the REST interface which is updated in the domain knowledge via the knowledge broker. The domain knowledge that exists in the resource description (RDF) format is updated by the knowledge broker simultaneously (Fig. 5*b*). The knowledge base in itself can be considered as a “raw” digital twin with potent information. The digital twin queries the domain knowledge base via SPARQL, an RDF query language. A sample query that initializes the components of the digital twin and locates them in the layout is shown in Fig. 5*c*. The digital twin (Fig. 6) makes use of this and a host of other information in the same manner to function and provide value to the entire business enterprise.

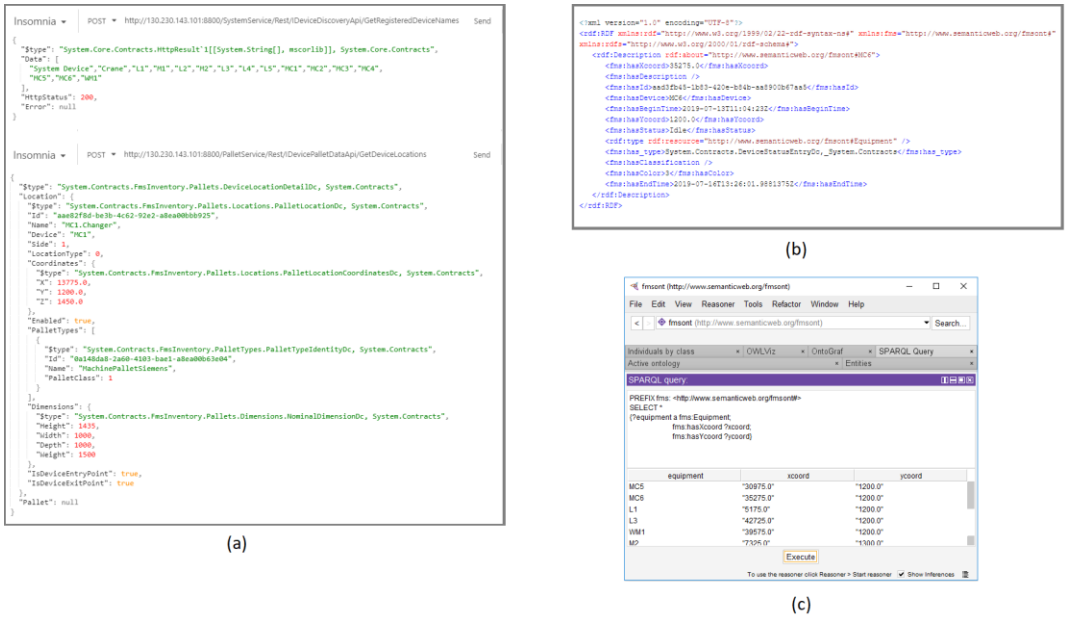

Figure 5. (*a*) Example RESTful information exchanged between Knowledge Broker and Production System (*b*) Machining Centre Description stored in RDF format in the Domain Ontology (c) SPARQL query to obtain coordinates of the devices in the production system

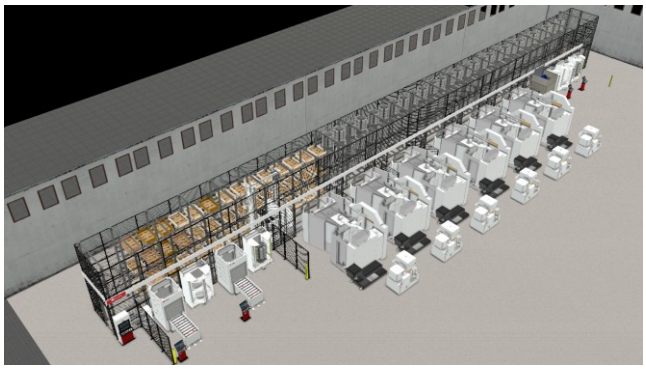

Figure 6. Digital Twin

DISCUSSION AND CONCLUSION

As the digitalization of the manufacturing enterprises gains momentum, the integration of its systems is considered to play a crucial role. By closing the loop between the ERP, MES and PLM the domain knowledge facilitates just this and the central repertoire of domain knowledge offers visibility and access that would help streamline manufacturing operations and help reduce redundant communication.

The use case presented a digital twin that exploits the architecture presented in the approach section. The use of an ontology expressed in OWL to represent domain knowledge can be justified by the fact that it offers high level of expressivity while allowing to model complex constructs typical of manufacturing systems. Inference of new knowledge (not in use-case) based on existing knowledge can also take place via inference engines using rule languages such as SWRL. Serializing it as RDF/XML helps existing and inferred knowledge to be queried by query languages such as SPARQL. The use of webservices further helps encapsulate business functionality while taking care of security concerns.

Moreover, such an architecture can have several other advantages. It provides scalability. As the physical asset scales, only the description of the services needs to be added or modified. If the scaling involves just increase in the quantity of existing components, the architecture allows for automatic discovery and updation in the domain knowledge. The knowledge broker further allows for easy access of all information stored in the knowledge base to any administrator or other users such as customers or suppliers.

With the data going vertically from the business planning and logistics system, to the field level we miss a big opportunity. Rather having a unified namespace for data across a common data model simplifies scalability and eliminates one-off integrations. Being able to "plug-in" applications and allowing applications to consumer necessary data greatly enhances the factories ability to adapt to change and maintenance, a key necessity for the factory of tomorrow.

REFERENCES

1. Agarwal, P. (2005). Ontological considerations in GIScience. *International Journal of Geographical Information Science, 19*(5), 501–536. doi: 10.1080/13658810500032321.
2. David, J., Lobov, A., & Lanz, M. (2018). Leveraging digital twins for assisted learning of flexible manufacturing systems. In: *2018 IEEE 16th International Conference on Industrial Informatics (INDIN)* (pp. 529–535). doi: 10.1109/INDIN.2018.8472083.
3. Diep, D., Alexakos, C., & Wagner, T. (2007). An ontology-based interoperability framework for distributed manufacturing control. In: *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA* (pp. 855–862). doi: 10.1109/EFTA.2007.4416869.
4. Fadel, F. G., Fox, M. S., & Gruninger, M. (2002). A generic enterprise resource ontology. In: *Proceedings of 3rd IEEE Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises* (pp. 117–128). doi: 10.1109/enabl.1994.330496.
5. Fielding, R. T., & Taylor, R. N. (2002). Principled design of the modern Web architecture. *ACM Transactions on Internet Technology, 2*(2), 115–150. doi: 10.1145/514183.514185.
6. Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J.-J., Nielsen, H. F., Karmarkar, A., & Lafon, Y. (2007). *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition).* Retrieved July 22, 2019, from https://www.w3.org/TR/soap12-part1/
7. Haldey, M. (2009). *Web Application Description Language: W3C Member Submission 31 August 2009.* Retrieved from https://www.w3.org/Submission/wadl/

8. Kalogeras, A. P., Gialelis, J. V, Alexakos, C. E., Georgoudakis, M. J., & Koubias, S. A. (2006). Vertical integration of enterprise industrial systems utilizing web services. *IEEE Transactions on Industrial Informatics, 2*(2), 120–128. doi: 10.1109/TII.2006.875507.
9. Lemaignan, S., Siadat, A., Dantan, J. Y., & Semenenko, A. (2006). MASON: A proposal for an ontology of manufacturing domain. In: *IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications (DIS'06)* (pp. 195–200). doi: 10.1109/DIS.2006.48.
10. Nach, H., & Lejeune, A. (2008). Implementing ERP in SMEs: Towards an ontology supporting managerial decisions. In: *2008 International MCETECH Conference on e-Technologies (mcetech 2008)* (pp. 223–226). doi: 10.1109/MCETECH.2008.11.
11. OWL – Semantic Web Standards. (n.d.). Retrieved July 22, 2019, from https://www.w3.org/OWL/
12. Sauter, T. (2005). Integration aspects in automation – a technology survey. In: *2005 IEEE Conference on Emerging Technologies and Factory Automation* (Vol. 2, pp. 9–263). doi: 10.1109/ETFA.2005.1612688.
13. Web Services Glossary § Web service, W3C. (2004). Retrieved July 21, 2019, from https://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/#webservice
14. Zoubeidi, M., Kazar, O., Mesbahi, N., & Benharzallah, S. (2014). Toward an architecture for the semantic integration in enterprise resource planning. In: *2014 International Workshop on Advanced Information Systems for Enterprises, IWAISE 2014* (pp. 45–50). doi: 10.1109/IWAISE.2014.10.