Dealing with data and software interoperability issues in digital factories*

Nicola Bicocchi[a], Giacomo Cabri[a], Federica Mandreoli[a,1] and Massimo Mecella[b]

[a] *Università degli Studi di Modena e Reggio Emilia*
<nbicocchi,gcabri,fmandreoli>@unimore.it
[b] *Sapienza Università di Roma*
massimo.mecella@uniroma1.it

Abstract. The digital factory paradigm comprises a multi-layered integration of the information related to various activities along the factory and product lifecycle manufacturing related resources. A central aspect of a digital factory is that of enabling the product lifecycle stakeholders to collaborate through the use of software solutions. The digital factory thus expands outside the actual company boundaries and offers the opportunity for the business and its suppliers to collaborate on business processes that affect the whole supply chain. This paper discusses an interoperability architecture for digital factories. To this end, it delves into the issue by analysing the main challenges that must be addressed to support an integrated and scalable factory architecture characterized by access to services, aggregation of data, and orchestration of production processes. Then, it revises the state of the art in the light of these requirements and proposes a general architectural framework conjugating the most interesting features of service-oriented architectures and data sharing architectures. The study is exemplified through a case study.

Keywords. smart factory, digital factory, interoperability framework, processes, services, data space

I. Introduction

The digital factory paradigm promotes the integration of product design processes, manufacturing processes, and general collaborative business processes across factories and enterprises. An important aspect is to ensure interoperability between the machines, products, processes, related products and services, as well as any descriptions of those. Accordingly, a digital factory consists of a multi-layered integration of the information related to various activities along the factory and product lifecycle manufacturing related resources [1]. With the increasing support of cyber-physical systems, smart electronics, sensors, robots, and embedded systems [2][3], data is constantly gathered enabling context-aware service integration [4] and management [5].

Moreover, in the near future, *(i)* factories and machines will be increasingly complex, *(ii)* dynamic situations will need to be managed during the whole product lifecycle, *(iii)* customers will be provided with personalized products (mass

* This work is partially founded by the EU H2020-RISE Project "FIRST: virtual Factory Interoperation suppoRting buSiness innovaTion" (734599).

1 Corresponding Author.

customization), *(iv)* human-centricity will be needed in order to increase flexibility, agility, and competitiveness, and *(v)* suppliers and customers will change frequently.

This vision of a nearby future for the digital factory is achievable through a general interoperability platform, in which people involved in the design and production processes are at the center, supported by the software tools in the implementation of their manufacturing, service, and business objectives. To this purpose, in this paper we propose a novel conceptual architecture based on three layers, *data, services, processes*, where processes and goal descriptions trigger the discovery of the services and data that best fit the expressed needs and their composition which is dynamic, autonomous and adaptive, in order to fully exploit limited human feedbacks.

The remainder of the paper is organized as follows: Section II introduces an example application scenario used throughout the paper, whereas Section III presents our conceptual architecture. Section IV compares our work with related ones and finally Section V concludes the paper by discussing possible future directions.

II. The muffin factory application scenario and related challenges

In order to introduce the different concepts and approaches proposed in this paper, we will use a case study. *MyMuffin* is a company operating in EU producing muffins willing to expand its business by allowing clients to buy muffins online. Clients can create their own muffins by picking pre-sets of ingredients and wait for its delivery[2].

The client orders box(es) (each one containing 4 muffins) online, by choosing among different possible variants, such as: *(a)* chocolate chips vs. blueberry vs. apricot bits vs. carrot bits vs. nothing as additional ingredient; *(b)* butter cream vs. hazelnut cream vs. icing sugar vs. nothing as topping; *(c)* yoghurt vs. honey vs. nothing in the dough. The client can also customize the colors of the baking paper (wrapping the single muffin) as well as the colors of the box.

The muffin factory collects orders and organizes batches of muffin doughs for production. As an example, if a client asks for 3 boxes of carrot muffins with yoghurt, icing sugar on top, pink baking paper, and another client for 2 boxes of carrot muffin with yoghurt, nothing on top, yellow baking paper, the same dough can be used for both. Clearly this scheduling service is based on the number of (and capacity of each) dough mixers, the stream of received orders, etc. The factory has a pool of dough mixers, of different capacity, and the fact that the number of different combinations is finite guarantees that such a scheduling can be performed.

When an order is received, in parallel to the dough preparation, the baking paper should be set-up as well. In addition to prepare a set of the specific requested color baking paper (the collection in batch can be performed as well), a QR-code should be printed on the baking paper and used as a unique identifier of the specific order. The correct identification of the single muffin is crucial for customization. After the dough has been prepared, the muffins are placed in the baking paper and sent to the oven

[2] MyMuffin is a fantasy company, but there are real successful examples of mass customization applied to food, cf. Mymuesli, a German company - https://en.wikipedia.org/wiki/Mymuesli. MyMuffin is an example of a small factory in which digital transformation can be applied in order to deeply modify production processes and business opportunities. Our work can be applied to such small factories as well as more complex ones, as in the automotive industry.

(connected to a QR code reader) for cooking. Muffins are cooked in batches of about 1000 items and the length of this step is equal for all of them.

After the baking has been performed, the cart is operated in order to route the different muffins to the right boxes, after putting the right topping, and then to the proper delivery station. Depending on the order (quantity and location of the customer) different delivery agents can be used.

Notably, flexibility is needed all along the process, e.g., the baking step may overcook some muffins, which therefore are not ready for the delivery and should be prepared again. This imply a communication with the delivery agent in order to skip the planned retirement and to set-up a new one (e.g., after a few hours or the day after) and also a re-scheduling of the mixers in order to re-introduce the preparation of the given dough.

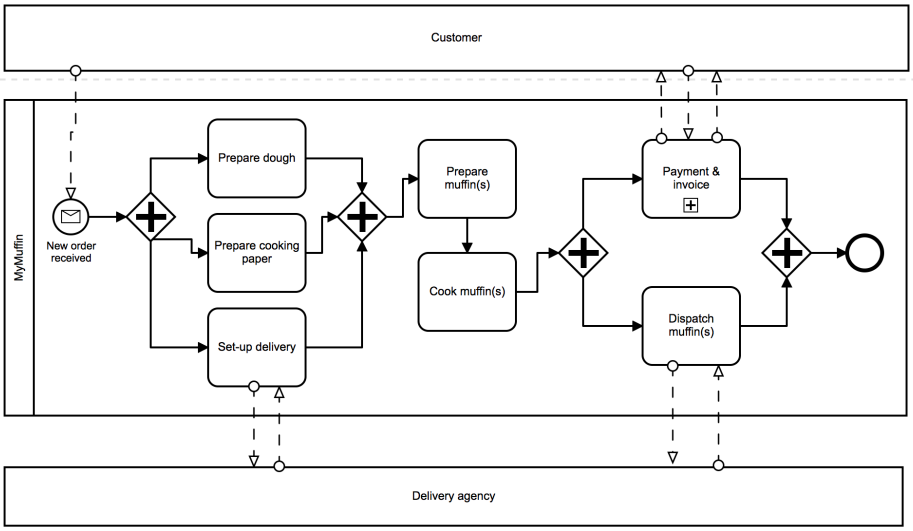

Figure 1. The process of MyMuffin. BPMN diagram, in which also public views of the delivery agency and the customer are shown as well (i.e., the whole supply chain)

III. A reliable interoperability architecture for virtual factories

One of the key issues in digital factories is to provide, manage and use the different services and data that are connected to the production processes. Manufacturing machines typically provide data about their status and services. These services are usually exploited at the digital factory level together with data and services coming from other departments, such as purchasing and marketing. We face heterogeneous situations: from the one hand, machines are from different vendors and, even if not proprietary, they are likely to adopt different standards and vocabulary, and data are managed by different systems as well; from the other hand, services can be provided at different levels of granularity, from very fine grained one (in terms of functionalities) to very coarse. As an example, the service of the oven may expose (simple fine grained) operations for `start()` and `stop()` itself, whereas the scheduling service exposes a (complex coarse grained) operation `schedule(listOfOrders)`:

`setOfMixerInstruction` which takes the list of received orders (not yet satisfied and up to the moment of invocation) and return the set of instructions to be given for the dough to the different mixers[3], The role of the digital factory is to *integrate* the different services and data and to combine them in order to make the whole process as efficient and competitive as possible in the achievement of the specific goals.

Another importation issue to be faced is the fact that the process can cover a space wider than the single factory (it supports a supply chain): usually a factory gets the raw material from suppliers and provide products or semi-finished products to customers, through delivery agents, requiring the corresponding services and data to integrate to each other or at least to be able to interact in a *scalable* and *flexible* way.

We propose to achieve this through a general three-layer interoperability framework, i.e., based on *processes, services, and data,* and assists users in the achievement of their objectives through the discovery of service and data flows that best fit the expressed requirements. In the following the three layers are detailed.

A. Process space layer - goal-oriented process specification

The top layer of the proposed architecture deals with the goals and the processes able to achieve such goals. In the MyMuffin example, some goals of the process are: [G1] *for each order, evade it within 36 hours (where evade means the muffins are packed and ready to be delivered)*; [G2] *for each order, the final delivery to the customer should be within 72 hours from the order*. The MyMuffin company adopts a process in which sub-goals might have been defined for specific parts (i.e., goals can in turn be decomposed in sub-goals), e.g., in order to achieve G1, it should be [G1.1] *muffin should not be overcooked*. Notably, MyMuffin would like to define, on the basis of such goals, specific KPIs – Key Performance Indicators, which qualify the QoS of the production process, e.g., the above 2 goals (i.e., G1 and G2) should be satisfied at least on 95% orders, in which the interval of observation is every week for orders received from Saturday 00:01 am till next Friday 23:59 pm.

Clearly goals and KPIs are defined over many aspects, including the interactions with external companies being part of the process (e.g., the delivery agents having as goal to employ maximum 24 hours from pick-up to delivery, and to keep a KPI of 95% satisfaction over the week).

B. Service space layer - dynamic service discovery and composition.

Starting from the goals and processes defined in the process layer, services must be dynamically composed to achieve the goal(s). In our example, we have different machines that can expose operations such as setting/increasing/decreasing the oven temperature, starting/stopping the dough mixer, etc., and providing data such as the duration of the dough preparation, the temperature of the oven, etc. OpenAPIs are exposed by such services in order to control, discover, and compose them in a dynamic way. Rich semantic descriptions of the services should be available in the interoperability platform, in order to support both the discovery of the services and their execution/invocation. The descriptions should include some keywords that identify the context of the service (e.g., “food”, “cooking”), the equipment (e.g.,

[3] `MixerInstruction` is an object which, for each mixer, details the ingredients to be put in the dough, and the quantity of dough to be loaded/produced in that mixer for the current batch.

"oven", "mixer"), the performed operation (e.g., "turn-on", "speedup"), and the parameters (e.g., "temperature", "speed").

With regard to the discovery phase, the semantic description is exploited to search for specific services without knowing their exact name and their syntax a priori. Semantic techniques can be exploited to find synonyms and keywords related to the words searched for in this phase. Searches can be performed either automatically by the process layer, in particular by the orchestration engine enacting processes, or by a human operator acting in the factory, which may be involved when needed (e.g., the adaptation techniques realized in the process layer fail, and a human intervention is needed in order to make the process progress) [6].

But the semantic descriptions can be exploited also in the composition phase. Being the composition dynamic, the platform must not only find but also exploit the needed service in an automatic way or providing an effective support to the human operator. To this purpose, the semantic description of the service parameters is needed in order exploit the meta-services of the data layer to adapt the client service invocation to the server syntax (see next subsection). Some proposals and examples of semantic service descriptions exist, such as in the SAPERE project [7] mentioned later.

The dynamism is useful to handle unexpected situations, often notified by a human operator; an example can be overcooked muffins, a case in which the courier must be notified to modify the shipment and a new set of muffins must be produced starting from the list of needed ingredients. To this purpose, the service `overcook():QRCode,type,num` is available in the platform and can be activated either by a monitoring facility or by human intervention. This service outputs the type (`type`) and number (`num`) of the overcooked muffins and the corresponding order (`QRCode`) and must be composed with two discovered services: one interacting with the courier (e.g., `shipment(URL)` with the courier Web service as input) and one activating the dosing machine (e.g., `dosing_machine(ingredient,quantity)` with ingredient and quantity as input). The composition (see Figure 1) requires the connection of the output with the input. Essentially, the composition connects the discovered services by making explicit the relationships between the involved service parameters. `?x`, `?y`, `?z`, `?h` are variables and the corresponding values must be discovered in the data space as they represent the input to the two services, `shipment` and `dosing_machine`.

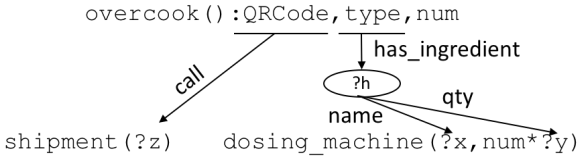

Figure 2. Service composition for the overcooked muffin

Clearly, the platform must also consider *failure* situations, such as oven out of work, refrigerator service not found, and so on. These issues require the frequent involvement of humans in the loop in order to deal with them in an effective way.

C. Data space layer - service-oriented mapping discovery and dynamic dataspace alignment.

Data are managed and accessed in a data space. The data space must be able to deal with a huge volume of heterogeneous data by autonomous sources and support the different information access needs of the service level. In particular, a large variety of data types should be managed at the dataspace level. According to the level of dynamicity, data can be static such as data available in traditional DBMSs but also highly dynamic like sensor data that are continuously generated. Moreover, it should accommodate data that exhibit various degrees of structures, from tabular data like relational data and CSV data to fully unstructured data like textual data. Finally, it should cope with the very diversified data access modalities sources offer, from low level streaming access to high level data analytics.

To this extent, the data modelling abstraction we adopt to represent the data space is fully decentralized, thereby bridging, on the one hand, existing dataspace models that usually rely on a single mediated view [6] and, on the other hand, P2P approaches for data sharing [8]. The dataspace is therefore a collection of heterogeneous data sources that can be involved in the processes, both in-factory and out-factory. Those data are either describing the manufactured products or the manufacturing processes and assets (material, machine, enterprises, value networks and factory workers) [9]. Each data source has its data access model that describes the kind of managed data, e.g., streaming data vs. static data, and the supported operators. As an example, sensed parameters such as temperature in the oven, temperature in the packing station, levels of the different ingredients, etc. are all streaming data needed in the dataspace of MyMuffin that can be accessed only through simple windowing operators on the latest values. On the other hand, supplier data can be recorded in a DBMS that offers a rich access model both for On Line Transaction Process (OLTP) operations and On Line Analytical Process (OLAP) operations.

Data representation relies on the graph modelling abstraction. This model is usually adopted to represent information in rich contexts. It employs nodes and labelled edges to represent real world entities, attribute values and relationships among entities. Figure 3 shows a small portion of the MyMuffin data space that can be used in case of overcooking. “Batches” is a data stream that reports the cooking status over time; “Orders” is the set of records storing the back orders made by client online and the corresponding QR-codes; “Recipes” is a semi-structured data set recording the recipes of the different kinds of muffins; “Yellow pages” is a web-based data source about the couriers and the related Web services. The oid’s in Figure 3, like oid_{101}, are object identifies and are used to collect together data referring to the same real-world entity. It is worth noting that graph data can be serialized in a triple base where each triple has the form *(s,p,o)*, where *s* is the source, *p* is the property, and *o* is the object.

The main problem the interoperability platform must cope with when dealing with data is data heterogeneity. Indeed, the various services gather data, information and knowledge from sources distributed over different stakeholders and external sources, e.g., the delivery agents and the Web. All these sources are independent, and we argue that a-priori agreements among the distributed sources on data representation and terminology is unlikely in large digital supply chains over several digital factories..

Data heterogeneity can concern different aspects: *(1)* different data sources can represent the same domain using different data structures; *(2)* different data sources can

represent the same real-world entity through different data values; *(3)* different sources can provide conflicting data. The first issue is known as *schema heterogeneity* and is usually dealt with through the introduction of mappings. Mappings are declarative specifications describing the relationship between a target data instance and possibly more than one source data instance. The second problem is called *entity resolution* (a.k.a. record linkage or duplicate detection) and consists in identifying (or linking or grouping) different records referring to the same real-world entity. Finally, conflicts can arise because of incomplete data, erroneous data, and out-of-date data. Returning incorrect data in a query result can be misleading and even harmful. This challenge is usually addressed by means of *data fusion* techniques that are able to fuse records on the same real-world entity into a single record and resolve possible conflicts from different data sources.

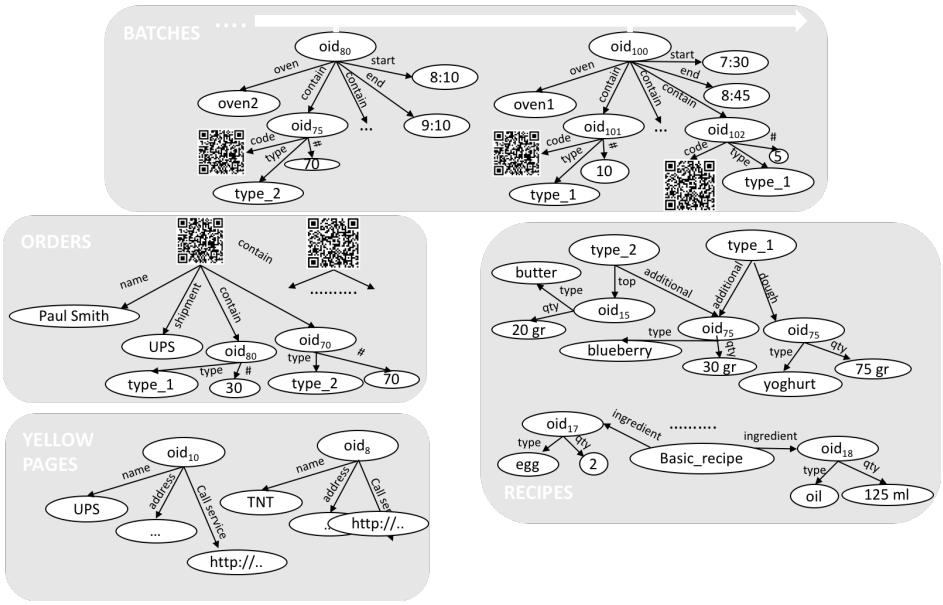

Figure 3. An excerpt of the MyMuffin data space

For instance, if the user is interested in reconstructing the current status of back orders, then it is necessary to fuse the data stored in the batches data source and the data stored in orders data source. In this case, entity resolution is necessary because the same muffin type of the same order is represented by different oid's (e.g., oid_{101} and oid_{80} or oid_{75} and oid_{70}) and data fusion is necessary because, when the information about the same muffin type in the same order are grouped together, there will be two edge symbols, i.e., "#", with different semantics, one representing the number of ordered pieces and the other one the number of cooked pieces.

Traditional approaches that address data heterogeneity propose to first solve schema heterogeneity by setting up a data integration application that offers a uniform interface to the set of data sources. This requires the specification of schema mappings that is a really time- and resource-consuming task entrusted to data curation specialists. This solution has been recognized as a critical bottleneck in large scale deeply

heterogeneous and dynamic integration scenarios, as digital factories are. A novel approach is the one where mapping creation and refinement are interactively driven by the information access needs of service flows and the exclusive role of mappings is to contribute to execute service compositions [10]. Hence, we start from a chain of services together with their information needs expressed as inputs and outputs which we attempt to satisfy in the dataspace. We may need to discover new mappings and refine existing mappings induced by composition requirements, to expose the user to the inputs and outputs thereby discovered for their feedback and possibly continued adjustments. Therefore, the service composition induces a data space orchestration that aims at aligning the data space to the specific service goals through the interactive execution of three steps: mapping discovery and selection, service composition simulation, feedback analysis. Mappings that are the outcome of this process can be stored and reused when solving similar service composition tasks.

Essentially, the data flow indicates that from each `QRCode` returned by the `overcook` service, *(i)* it should be derived the Web service to interact with the delivery agent/courier, whereas *(ii)* from the type of the overcooked muffin it should be derived the list of ingredients together with the required quantity as input to the dosing machine.

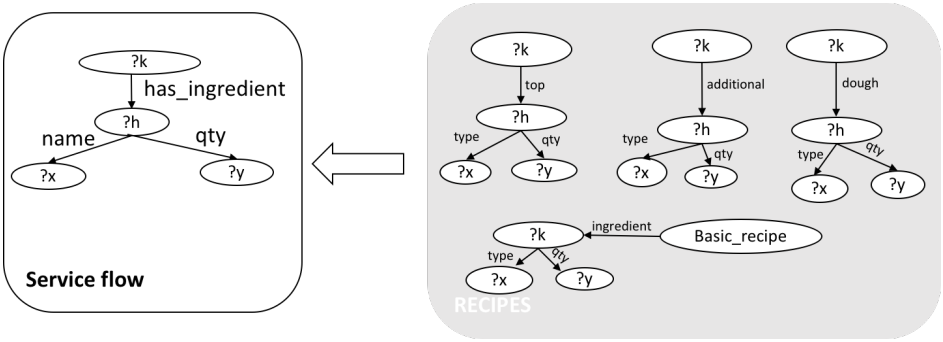

Figure 4. Mapping discovery process

Therefore, mapping discovery leads to two mappings whose targets are `(QRCode,call,?z)` and `(type,has_ingredient,?h), (?h,name,?x), (?h,qty,num*?y)`. A plausible output to the mapping discovery for the second mapping is shown in Figure 3. This mapping involves the "Recipes" data source, only, and provides all the ingredients of the recipe of the type of the given overcooked muffins. If some muffins of type `type_1` are overcooked then `?k=type_1` and the input to the dosing machines will be `(yoghurt,75gr), (blueberry,30gr), (egg,2)`, etc. Notice that the discovery of such a mapping most likely needs human intervention because, given a muffin type, some alternatives are available to get to the corresponding ingredients and the addition of the basic recipe ingredients is not so obvious.

IV. Related Work and Concluding Remarks

The composition of resource services, as an approach aimed at improving the efficiency of service selection and utilization, is widely adopted at present in

manufacturing [11]. Supported by recent technologies such as service oriented computing, the Internet of Things (IoT) and mobile platforms, the virtualization of manufacturing processes is rapidly expanding [12][13]. A variety of methods for resource composition have recently been proposed.

IoT technology has been applied to the problem of service composition for improving both resource selection and utilization [14]. Though the composition of resource services is important, cross-organization is seldom considered in such an environment. How a cross-organizational resource configuration impacts performance is discussed in [15]. Quality of service (QoS)-aware service composition in cloud manufacturing (CMfg) systems has been proposed. As an example, the system proposed in [16] allows a free combination of multiple functionally-equivalent elementary services into a synergistic elementary service group to perform each subtask collectively, thereby improving the overall QoS.

Framework-based methods have been also used for the composition of resources. In [14], a configurable information service platform is proposed for the development of IoT-based applications, providing an information support base for both data integration and intelligent interaction in the product lifecycle. Based on an abstract information model, information encapsulating, composing, discomposing, transferring, tracing, and interacting in Product Lifecycle Management (PLM) can be carried out. Combining ontologies and representational state transfer (REST)-ful services, the platform provides an information support base both for data integration and intelligent interaction.

The SAPERE project [8] is a general coordination framework aimed at facilitating the decentralized execution of self-organizing and self-adaptive services. It conceptually models a service ecosystem as a virtual environment. The interactions between services take place by applying a limited set of basic interaction laws, and typically take into account the spatial and contextual relationships between services.

To the best of our knowledge, the proposed approach is different from the alternative ones for the following reasons: *(i)* it pursues a global approach that starts from the processes and arrives at data; *(ii)* it puts humans in-the-loop of PLM without requiring heavy manual intervention; *(iii)* it relies on dynamic orchestration of services and data to align them to the processes; *(iv)* it supports personalized paths towards process goals.

Despite the encouraging approach, there are still several open research issues to be addressed in the future to realize the proposed architecture such as: *(i)* the definition of a way to describe goals related to the process; *(ii)* the definition of a way to semantically describe the services in the system; *(iii)* the definition of a way to describe the result of the service composition.

References

[1] Chungoora, N., Young, R. I., Gunendran, G., Palmer, C., Usman, Z., Anjum, N. A., & Case, K. (2013). A model-driven ontology approach for manufacturing system interoperability and knowledge sharing. Computers in Industry 64:4, pp. 392-401.

[2] Da Xu, L., He, W., & Li, S. (2014). Internet of things in industries: A survey. IEEE Transactions on industrial informatics, 10:4, 2233-2243.

[3] Monostori, L. (2014). Cyber-physical production systems: Roots, expectations and R&D challenges. Procedia CIRP, 17, pp. 9-13.

[4] Davis, J., Edgar, T., Porter, J., Bernaden, J., & Sarli, M. (2012). Smart manufacturing, manufacturing intelligence and demand-dynamic performance. Computers & Chemical Engineering, 47, pp. 145-156.

[5] Bi, Z., Da Xu, L., & Wang, C. (2014). Internet of things for enterprise systems of modern manufacturing. IEEE Transactions on industrial informatics, 10(2), pp. 1537-1546.

[6] Marrella, A., Mecella, M., & Sardiña, S. (2017). Intelligent Process Adaptation in the SmartPM System. ACM Transaction on intelligent systems, 8(2).

[7] Castelli, G., Mamei, M. Rosi, A., & Zambonelli F. (2015). Engineering pervasive service ecosystems: The SAPERE approach. ACM Transactions on autonomous and adaptive systems, 10(1).

[8] Penzo, W., Lodi, S., Mandreoli, F., Martoglia, R., Sassatelli, S. (2008). Semantic peer, here are the neighbors you want!. Proceedings of ACM EDBT.

[9] EFFRA Factories 4.0 and Beyond Recommendations for the work programme 18 - 19 - 20 of the FoF PPP under Horizon 2020, Online: http://www.effra.eu/sites/default/files/factories40_beyond_v31_public.pdf

[10] Mandreoli, F. (2017). A Framework for User-Driven Mapping Discovery in Rich Spaces of Heterogeneous Data. Proc. 16th OTM Conferences (2), pp. 399-417.

[11] Li, H., Chan, K. C. C., Liang, M., & Luo, X. (2016). Composition of Resource-Service Chain for Cloud Manufacturing. IEEE Transactions on industrial informatics, 12(1).

[12] Cai, H., et al. (2014). IoT-based configurable information service platform for product lifecycle management. IEEE Transactions on industrial informatics, 10(2), pp. 1558-1567.

[13] Tao, F., et al. (2014). IoT-based intelligent perception and access of manufacturing resource toward cloud manufacturing. IEEE Transactions on industrial informatics, 10(2), pp. 1547-1557.

[14] He, W., & Xu, L. D. (2014). Integration of distributed enterprise applications: A survey. IEEE Transactions on industrial informatics, 10(1), pp. 35-42.

[15] Edgar, T. L., Chiotti, O., & Villarreal, P. D. (2014). Software agent architecture for managing inter-organizational collaborations. Journal of applied research and technology, 12(3), pp. 514-526.

[16] Bo, L., & Zhang, Z. (2017). QoS-aware service composition for cloud manufacturing based on the optimal construction of synergistic elementary service groups, International Journal of Advanced Manufacturing Technology, 88.9-12, pp. 2757-2771.