Dynamic digital factories for agile supply chains: An architectural approach

Abstract

Digital factories comprise a multi-layered integration of various activities along the factories and product lifecycles. 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 company boundaries and offers the opportunity to collaborate on business processes affecting the whole supply chain.

This paper discusses an interoperability architecture for digital factories. To this end, it delves into the issue by analysing the key requirements for enabling a scalable factory architecture characterised by access to services, aggregation of data, and orchestration of production processes. Then, the paper revises the state-of-the-art w.r.t. these requirements and proposes an architectural framework conjugating features of both service-oriented and data-sharing architectures. The framework is exemplified through a case study.

Keywords: smart factory, digital factory, interoperability framework, process, service, data space

1. Introduction

Production processes are nowadays fragmented across different companies and organized in global multi-tier supply chains. This is the result of a first wave of globalization that, among the various factors, was enabled by the diffusion of Internet-based Information and Communication Technologies (ICTs) at the beginning of the years 2000. The recent wave of new technologies possibly leading to the fourth industrial revolution – the so called *Industry 4.0* – is further multiplying opportunities. Accessing global customers opens up great opportunities for firms, including small and medium enterprises (SMEs), but it requires the ability to adapt to different requirements and conditions, volatile demand patterns and fast changing technologies.

Supply chains are required to be more and more *agile*, where agility is defined as a combination of responsiveness and resilience. More specifically, *responsiveness* concerns the ability to adapt to changes in the demand, provide customers with personalized products (mass customization), quickly exploit temporary or permanent advantages and keep their competitive edge, while *resilience* concerns the ability to react to disruptions along the supply chain. The resulting agile supply chains will be able to successfully adapt to an evolving and uncertain business context in terms of both demand (customization, variability, unpredictability) and supply (new components, uncertainty in the supplies, bottlenecks and risks) taking into account not only the single organization but the entire value chain.

Our aim is to investigate methods and techniques for enhancing global multi-tier supply chains by addressing the methodological issues of how to apply digital technologies into existing supply chains and proposing a reference architecture.

Digital factory is a key paradigm to this end, as it aims at using digital technologies to promote the integration of product design processes, manufacturing processes, and general collaborative business processes across factories and enterprises [1, 2]. An important aspect of this integration is to ensure *interoperability* between machines, products, processes, 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 related resources.

At the same time, leading institutions and firms in Europe, and specifically in Germany, have developed and published the Reference Architecture Model Industry 4.0 (RAMI 4.0) [3]. It describes the fundamental aspects of Industry 4.0 and aims to achieve a common understanding of what standards and use cases are required for Industry 4.0. Both the technological principles of digital factories and the RAMI 4.0 architectural principles are of particular importance for our purposes. However, there are still open challenges to be addressed in order to meet the requirements of agile supply chains.

In the following, we first introduce a case study scenario providing an exemplification of the main factors of an agile supply chain. Then, we overview the key contributions of this work.

1.1. The muffin factory application scenario

MyMuffin is a company operating within the EU producing muffins and 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 delivery[1]. A 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

[1] 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.

vs. icing sugar vs. nothing as topping; *(c)* yogurt 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 yogurt, icing sugar on top, pink baking paper, and another client for 2 boxes of carrot muffins with yogurt, nothing on top, yellow baking paper, the same dough can be used for both orders. 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. 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 requested paper baking cases, a QR-code should be printed on each of them and used as a unique identifier of the specific order. The identification of the single muffin is crucial for customization. After the dough has been prepared, the muffins are placed in the baking paper cases and sent to the oven (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, different delivery agents can be used. Notably, agility 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 shipping and to set-up a new one and also a re-scheduling of the mixers in order to re-introduce the preparation of the damaged dough.

Figure 1 shows the process represented in Business Process Model and Notation (BPMN), cf. `http://www.bpmn.org/`. The reader not knowleadgeable about BPMN can read a short introduction about it in Section 2.1, where a detailed explanation of the graphical notation adopted in the figure is also provided.

1.2. Paper contribution and outline

The main contribution of this paper is to provide a methodological and technological support to agile supply chains in the Industry 4.0 context. To this end, it sets forth an architectural framework that leverages RAMI 4.0 and addresses the methodological issue of making RAMI 4.0 capable of enabling agility in supply chains.

The proposed architectural framework enables interoperability through a three-layered architecture where business processes and goal descriptions trigger the discovery of the needed services and data, and their composition in a dynamic, autonomous and adaptive fashion.

The rest of the paper is organized as follows: Section 2 provides an overview of the state of the art, Section 3 is the core section that presents the RAMI 4.0-based architectural framework, and finally Section 4 concludes the paper.

2. Background and related work

As pointed out in [4], pre-requisites for digital platforms to thrive in a manufacturing environment include the need for agreements on industrial communication interfaces and protocols, common data models and semantic interoperability. Currently, automated production plants, in fact, routinely employ thousands of devices from hundreds of vendors [5]. Furthermore, the growing importance of cooperation among organizations, encourages to dynamically establish inter-organisational interoperation.

In this situation, interoperability becomes a relevant challenge. Service Oriented Architectures (SOAs), Internet-of-Things (IoT) technologies, and open standards for device classification and discovery have been introduced to mitigate these issues [6, 7]. The most prominent examples of these trends are described in the following, whereas a detailed survey of the field is presented in [8].

Overall, despite the recent efforts aimed at the digitalisation of manufacturing, current approaches are still lacking in one or more of the following dimensions: *(i)* they still do not pursue a seamless integrated approach, which starting from processes arrives to data; *(ii)* they do not keep humans in-the-loop of product lifecycle management; *(iii)* they do not support in-process dynamic orchestration of services and data; *(iv)* they do not support alternative or personalised paths towards process goals.

2.1. Process management

Business Process Management (BPM) is a well-established discipline that deals with the identification, discovery, analysis, (re-)design, implementation, execution, monitoring, and evolution of business processes [9]. A business process is a collection of related events, activities, and decisions that involve a number of actors and resources that collectively lead to an outcome that is considered of value. Examples of business processes include order-to-cash, procure-to-pay, application-to-approval, claim-to-settlement, or fault-to-resolution.

To support business processes at an operational level, a BPM system (BPMS) can be used. As opposed to data- or function-centered information systems, a BPMS separates process logic from application code and, thus, provides an additional architectural layer. Typically, a BPMS provides generic services necessary for operational, software-enabled business process support, i.e., for process modeling, process execution, process monitoring, and user interaction (a.k.a. worklist management). When using a BPMS, software-enabled business processes are designed in a top-down manner, i.e., process logic is explicitly described in terms of a process model providing the schema for process execution. The BPMS is responsible for instantiating new process instances, for controlling their execution based on the process model, and for completing them. The progress of a process instance is typically monitored and traces of execution are stored in an event log and can be used for process mining

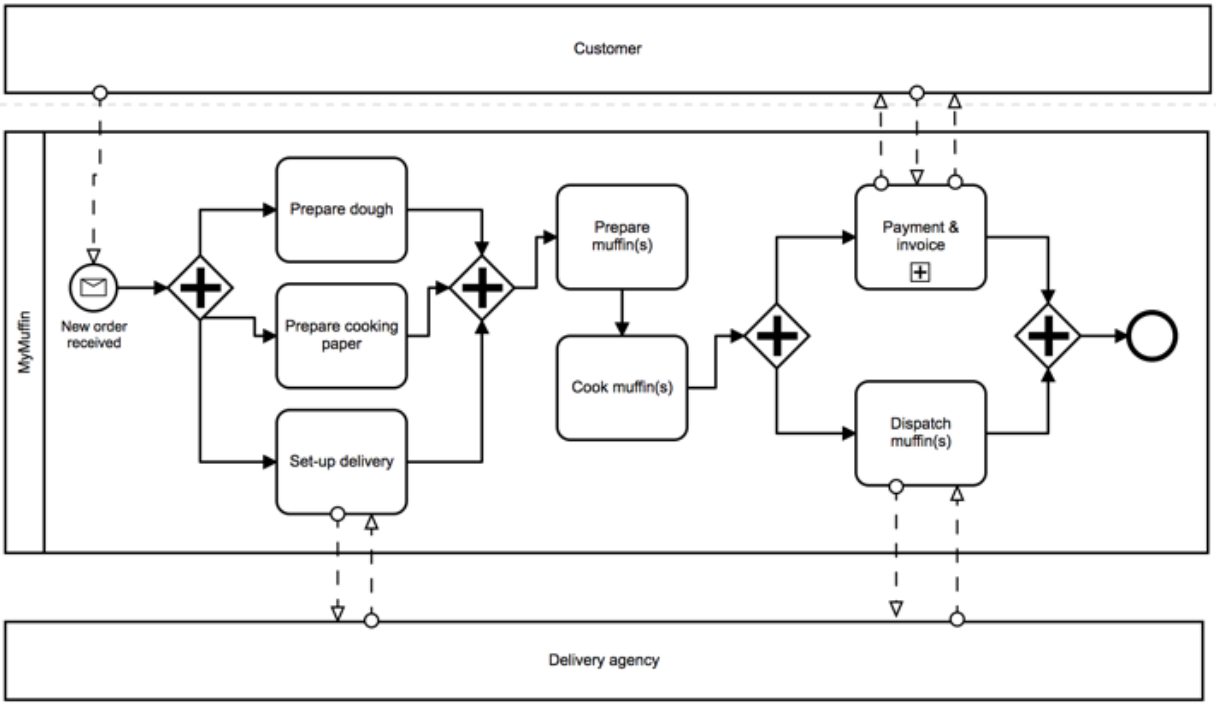

Figure 1: The process of MyMuffin. BPMN diagram, in which also public views of the delivery agency and the customer are shown as well (i.e., the whole supply chain).

[10], e.g., the discovery of a process model from the event log or for checking the compliance of the log with a given process model.

So far, the predominant paradigm to develop operational support for business processes has been based on the *Model-Enact* paradigm, where the business process has been depicted as a (graphical) process model, which then could be executed by a BPMS. This largely follows a top-down approach and is based on the idea of a central orchestrator that controls the execution of the business process, its data, and its resources.

With the emergence of IoT, the existing Model-Enact paradigm is challenged by the *Discover-Predict* paradigm; it can be characterized as a bottom-up approach where data is generated from devices sensing their environment and producing events. Sensor data must be then aggregated and interpreted in order to detect activities that can be used as input for process mining algorithms supporting decision-making [11].

BPMN is the standard for business process modeling that provides a graphical notation for specifying business processes in a business process diagram (BPD), based on a flowcharting technique. A diagram is constructed with a limited set of graphical elements explained below, by using Figure 1 as an example.

- Events, represented with circles, denote something that happens (compared with an activity, which is something that is done). Icons within the circle denote the type of event (e.g., an envelope representing a message, or a clock representing time). In the example in Figure 1, the start event of the process is when there is a *New order received*, and the process terminates when the flow reaches the bold-border circle.

- Activities, depicted as rounded rectangles, represent the single units of work. In our case study, they are *Prepare dough*, *Prepare cooking paper*, *Set-up delivery*, *Prepare muffin(s)*, *Cook muffin(s)*, *Dispatch muffin(s)* and *Payment & invoice*. Notably *Payment & invoice* is a sub-process, indicated by a plus sign against the bottom line of the rectangle, as it represents a compound activity, to be possibly detailed in its own diagram.

- Gateways, depicted with diamond shapes, determine forking and merging of paths. Exclusive gateways (showing an X inside the diamon) are used to create alternative flows in a process, as only one of the paths can be taken; parallel gateways (showing a + inside the diamond) are used to create parallel paths without evaluating any conditions. In the example, only parallel gateways are used, to mean that *Prepare dough*, *Prepare cooking paper* and *Set-up delivery* are all performed in parallel, then the flow is synchronized, and after some more activities performed sequentially, again *Dispatch muffin(s)* and *Payment & invoice* are performed in parallel.

- Connections are used to connect activities/events and gateways. *(i)* A sequence flow is represented with a solid line and arrowhead, and it shows in which order the activities are performed. As an example, *Prepare muffin(s)* is sequentially followed by *Cook muffin(s)*. *(ii)* A message

flow is represented with a dashed line, an open circle at the start, and an open arrowhead at the end. It tells us what messages flow across organizational boundaries (i.e., between pools – see further). A message flow can never be used to connect activities or events within the same pool. In the example, *Customer* sends a message to *MyMuffin* to start the process, messages are exchanged as well during the sub-process *Payment & invoice*. Analogously, messages are exchanged between *MyMuffin* and the *Delivery agency* during the activities *Set-up delivery* and *Dispatch muffin(s)*.

- Pools and lanes are used to represent participants in a process. In particular, each separate organization is represented as a pool (rectangle), as *Customer*, *MyMuffin* and *Delivery agency* in the example. A pool can contain one or more lanes, when the designer/modeler may want to organise and categorise activities according to a function or role within the same organization. A pool can be open (i.e., showing internal details, as *MyMuffin* in the example) when it is depicted as a large rectangle showing one or more lanes, or collapsed (i.e., hiding internal details, as *Customer* and *Delivery agency* in the example) when it is depicted as an empty rectangle stretching the width or height of the diagram. Notably, no specific functions/roles are depicted for *MyMuffin*, so no lanes are represented. When an organization is depicted as a collapsed pool, it is said to offer a *public view* of its processes, to mean that no internal details are exposed. In the example, *MyMuffin*, which is the subject of investigation, is completely modeled, whereas only the public views of *Customer* and *Delivery agency* are represented (i.e., their presence and the exchanged messages).

In the digital factory context, most of the works have been so far devoted to modeling issues, and specifically in the identification of suitable abstractions and modeling approaches and tools combining well-known standards with the specificities of digitial factories, e.g., [12, 13]. Recently, the focus is shifting toward dynamicity during run-time, e.g., [14], in order to have automatic adaptation of production processes.

2.2. Service Oriented Architectures

A Service Oriented Architecture (SOA) is a valuable candidate for supporting integration among multiple conceptual layers and making distributed systems open and interoperable. Large enterprises promoted their use in manufacturing since late 90s [15]. A SOA offers the potential to provide the necessary system visibility and device interoperability in complex automation systems subject to frequent changes. A SOA can be considered as an architectural paradigm defining mechanisms to publish, find and compose services adopting loose coupling principles and open standards. It provides with technologies, methods and tools that can enhance interoperability by decoupling functionalities and their implementations. As a consequence, the transparency of the entire structure is increased, thus making the SOA paradigm particularly applicable in environments where reconfigurability is highly desirable.

Several recent EU research and innovation projects, such as SOCRADES [16], SODA [17], SIRENA [18], have demonstrated the feasibility of embedding web services at the device level and integrating these devices with a Manufacturing Execution System (MES) and Enterprise Resource Planning (ERP) system, at the upper levels of an enterprise architecture [19]. More in detail, SOA have been investigated for studying cross-organisational resource configuration [20], resource selection and utilisation [21] and Product Lifecycle Management (PLM, [22]). In [20], an agent-based software architecture for managing inter-organizational collaborations is proposed. A Colored Petri Net model specifying the role, which an organization fulfills in a collaborative process, is used to carry out the behavior of the agent representing the organization. In [21], it is proposed a solution for constructing a supply-chain information exchange platform. It adopts an heterogeneous data exchange engine and a data exchange agent to perform certain service functions such as end-to-end data exchange. In [22], a cloud-based framework capable of accommodating any kind of services and providing session control is proposed. More specifically, the framework enables services to collaborate with any combination of other services on the framework.

2.3. IoT technologies

The decentralised execution of self-organising and self-adaptive services has been recently discussed. In particular, the SAPERE project [23] conceptually models a service ecosystem as a virtual environment (e.g., a virtual factory). 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.

IoT technology has been applied to the problem of service composition for improving both resource selection and utilisation [24]. More in detail, a configurable 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, by combining ontologies and RESTful services. Based on an abstract information model, information encapsulating, composing, discomposing, transferring, tracing, and interacting in PLM can be carried out.

Though the composition of resource services is important, cross-organisation is seldom considered in such an environment. How a cross-organisational resource configuration impacts performance is discussed in [25].

Quality of service (QoS)-aware service composition in cloud manufacturing (CMfg) systems has also been proposed. As an example, the system proposed in [26] 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. To deal with the increasing computing complexity of the optimisation model, an algorithm, named matrix-coded genetic algorithm with collaboratively evolutional populations, has been designed.

A similar approach is discussed in [27]. A genetic algorithm was used to achieve global optimisation with regard to service level agreements – SLAs. Moreover, service clustering was

used for reducing the search space of the problem, and association rules were used for a composite service based on their histories to enhance service composition efficiency.

2.4. Asset description, classification and discovery

Device integration makes data and functionalities of devices available throughout the entire automation system in ways that support association, integration, data exchange, and possibly semantic descriptions. Currently, the most widespread and relevant technologies include Electronic Device Description Language (EDDL), Field Device Tool (FDT)/Device Type Manager (DTM) and Field Device Integration (FDI).

With FDI, a technology has been developed that combines the advantages of FDT with those of EDDL in a single, scalable solution. FDI considers the various tasks over the entire lifecycle for both simple and the most complex devices, including configuration, commissioning, diagnosis and calibration [28]. Globally leading control system and device manufacturers, such as ABB, Emerson, Endress+Hauser, Honeywell, Invensys, Siemens and Yokogawa, along with the major associations FDT Group, Fieldbus Foundation, HART Communication Foundation, OPC Foundation, PROFIBUS PROFINET International, are supporting the development of the FDI together.

In most scenarios, taxonomies are usually adopted as common ground for semantic interoperability. Classifying products and services with a common coding scheme facilitates commerce between buyers and sellers and is becoming mandatory in the new era of electronic commerce. Large companies are beginning to code purchases in order to analyse their spending. Samples of taxonomy including the description and classification of manufacturing assets and services are: eCl@ss, UNSPSC, and MSDL [29].

Nonetheless, this approach to semantic interoperability cannot be employed in the considered agile application scenarios. Indeed, most coding systems today have been very expensive to develop and do not rapidly adapt to context changes. The effort to implement and maintain these systems usually requires extensive utilization of resources, over an extended period of time. Additionally, maintenance is an on-going and expensive process. Another problem is that company's suppliers not necessarily and always do adhere to the coding schemes of their customers, if any.

With the increasing number of assets, service discovery becomes an integral part of digital factories. Service discovery provides a mechanism which allows automatic detection of services offered by any component in the system. The objective of a service discovery mechanism is to develop a highly dynamic infrastructure where requestors would be able to seek particular services of interest, and service providers offering those services would be able to announce and advertise their capabilities. Furthermore, service discovery should minimize manual intervention and allows the system to be self-healing by automatic detection of services which have become unavailable. Once services have been discovered, devices in the system could remotely control each other by adhering to some standard of communication. Over the past years, many organizations and major software vendors have designed and developed a large number of service discovery protocols such as SLP, Jini, UPnP and UDDI.

2.5. Data sharing and interoperability

A global multi-tier supply chain necessarily requires the interconnection among different and often heterogeneous information systems. From the perspective of data management, the main issue is to effectively manage heterogeneity in a dynamic context while preserving the autonomy of the data sources. Indeed, the different information systems offer data, information and knowledge from sources distributed over different stakeholders. All these sources are independent, making thus a-priori agreements unlikely.

Given a collection of disparate and distributed data sources, the main objective is often to provide a unified view (i.e., *data integration*) or to enable the exchange of data among them (i.e., *data exchange*) [30]. In this context, the main difficulty lies in the fact that there is no agreement on the adopted data management systems, data models and languages, the vocabularies and structures used to describe the data (often denoted as *schema*) and the semantics of data values. Relationships between the data exposed by heterogeneous information systems are usually expressed through mappings that are declarative specifications spelling out the relationship between a target data instance and possibly more than one source data instance [31].

During the last two decades, many aspects concerning data sharing and interoperability have been studied including data management abstraction and architectures. The most interesting proposals that can be exploited to devise an interoperability platform for digital factories supporting agile and global multi-tier supply chains are *(i)* dataspaces, *(ii)* peer data management systems, and *(iii)* polystores.

A dataspace [32] is an abstraction for data integration allowing the coexistence of heterogeneous data sources by providing basic functionalities over all data sources, regardless of how integrated they are. The goal is to reduce the effort required to set up a data integration system by relying on existing mapping generation techniques, and to improve the system in a *pay-as-you-go* fashion. Dataspace principles can be thus exploited to manage dynamic situations. Moreover, the interaction with end users is the distinctive element of some pay-as-you-go approaches for dataspace systems.

A peer data management systems (PDMS) [33] is defined as a set of autonomous peers exposing data and the related schema and a set of schema mappings. A PDMS therefore is a distributed data integration system providing transparent access to heterogeneous databases without resorting to a centralized schema. Instead of imposing a uniform query interface over a mediated schema, a PDMS let the peers define their own schemas and the consequent reformulation of queries through mappings relating schemas.

Polystore systems [34, 35] have been recently proposed as a flexible architectural solution to data sharing and interoperability pursuing the *one-size-does-not-fit-all* philosophy. They enable query processing over heterogeneous stores while guaran-

teeing full source autonomy, just-in-time transparent data transformation and support to multiple query interfaces.

3. Enabling interoperability in digital factories

The approach undertaken in this work is based on RAMI 4.0 (cf. Figure 2). RAMI is a three dimensional reference architectural framework in the manufacturing industry domain developed in Germany by leveraging EU initiatives and guidelines[2][3].

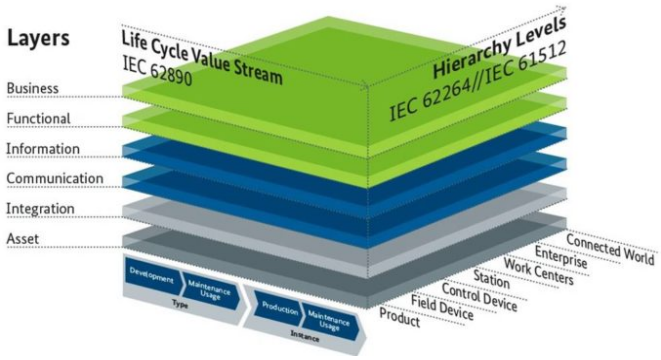

Figure 2: RAMI 4.0. A three dimensional reference architectural framework in the manufacturing industry domain.

We leverage RAMI 4.0 as the reference architectural framework describing how to apply digitalization technologies into existing supply chains to make them agile. According to RAMI 4.0, data is the bridge towards digitalization and is described in the integration, communication and information layers. In global multi-tier supply chains, data characteristics are largeness, distribution and heterogeneity. For instance, machines equipped with IoT sensors continuously produce data streams, OLTP data are available in DBMSs, OLAP data are available in data warehouses, digital manuals are stored in repositories, and so on. To deal with these data features, data is organized in a dataspace of data sources that can exchange data through mappings. The dataspace adhere to the polystore architectural model supporting dynamic configurations (i.e, data sources going in and out the system).

On top, at the functional level, different kinds of services are provided to get information and to perform actions on the manufacturing parts of system (e.g., producing and assembling machines) as well as to enable interoperability with different actors of the supply chain (e.g., order management, warehouse management). Open APIs are exposed by services in order to control, discover, and compose them in a dynamic way. Rich semantic descriptions of the services should be available in order to support both the discovery of the services and their execution/invocation. This lays the foundations to achieve higher-level goals defined at the business level.

[2]Cf. https://www.plattform-i40.de/I40/Navigation/EN/InPractice/Online-Library/online-library.html

[3]Cf. https://ec.europa.eu/futurium/en/system/files/ged/a2-schweichhart-reference_architectural_model_industrie_4.0_rami_4.0.pdf

At the business level, in fact, business process specifications must be able to capture not only orchestrated processes - which are bounded inside a single organization - but also choreographed processes which spans among different organizations, as a supply chain definition requires. Moreover, agility in the business processes can be achieved by shifting from the typical activity-centric process modeling to an artifact-centric modeling. This allows to model agile business processes with more emphasis to the goal to be achieved (i.e., the status to be reached) [36]. By defining several goals, with different degrees of completeness, the business process model is able to support a resilient and responsive environment, as the involved parties can tune their efforts to reach one of the goals, that is not necessarily the best one. Decisions on the goal to be achieved are driven by the available data [36].

One of the key issues to support agile supply-chains 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. 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; on the other hand, services can be provided at different levels of granularity, from very *fine grained* one (in terms of functionalities) to very *coarse grained*. As an example, the service of the oven may expose (simple, fine grained) operations for `start()` and `stop()`, whereas the scheduling service exposes a (complex, coarse grained) operation `schedule(listOfOrders): setOfMixerInstructions` which takes the list of received orders and return the set of instructions to be given for the dough to the different mixers. 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 important 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 envision a dynamic framework capable of assisting users through the discovery of service and data flows that best fit the expressed requirements and their evolution. The overall picture of the resulting RAMI 4.0-based architectural framework and the involved technological solutions are shown in Figure 3 . In the following the three layers are detailed.

3.1. 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 My-Muffin 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.

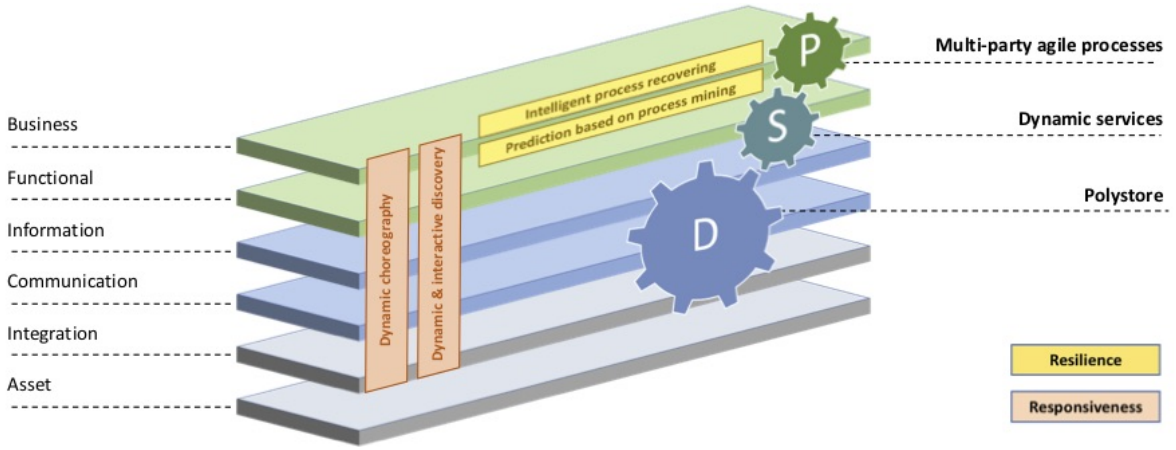

Figure 3: The enhanced RAMI 4.0. A dynamic framework capable of assisting users through the discovery of service and data flows that best fit the expressed requirements and their evolution.

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] muffins 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 on weekly basis. 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% respected over the week).

As an example of agility, we can imagine that in a given day, some muffins get overcooked due to an error in the oven. This means that the goal [G1.1] is not achieved. In such a case, the digital platform will operate in order to re-arrange the process to achieve the goal. Through automated planning techniques, as the one adopted in SmartPM [14], the process can be modified as shown in Figure 4. In particular, after the original activities *Prepare muffin(s)* and *Cook muffin(s)* (cf. Figure 1), new activities are introduced, in order to *Select alternative cooking service*, as a local bakery nearby MyMuffin that offers the availability of the oven; then, analogously to the original process, *Prepare dough* and *Prepare cooking paper* are performed, the muffins are moved and finally are received freshly cooked (cf. *Move muffin(s)* and *Receive freshly cooked muffin(s)*). Finally the process prosecutes as the original one. Notably, this is only one of the possible adaptations, the more complex as it re-arranges the process; in the example, it is used if simpler solutions are not possible in the given situations. We will see later that other solutions at the underlying levels are possible, depending on the specific situation.

3.2. Service space layer - service discovery and composition

Starting from the goals and processes defined in the process layer, services must be dynamically composed to achieve 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 and providing related data by means of OpenAPIs. Rich semantic descriptions of the services should be available, in order to support both discovery and service execution. The descriptions should include some keywords that identify the context of the service (e.g., "food", "cooking"), the equipment (e.g., "oven", "mixer"), the performed operation (e.g., "turn-on", "speedup"), and the parameters (e.g., "temperature", "speed").

With regard to the discovery phase, semantic descriptions are exploited to search for specific services without knowing their exact names 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 or by human operators which may be involved when needed (i.e., the adaptation techniques realized in the process layer fail, and a human intervention is needed in order to make the process progress).

Semantic descriptions can be used in the composition phase as well. Being the composition dynamic, the platform must not only find, but also use, the needed service or eventually provide support to human operators. To this purpose, the semantic description of the service parameters is needed in order to exploit the functionalities 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 have already been proposed [23].

The dynamism is useful to handle unexpected situations, often notified to a human operator. We report a couple of examples: in the former, an unexpected event causes an internal reorganization of the tasks; in the latter, an unexpected event deserves the interaction with an external actor.

The first example concerns oven performance. It may happen that the oven does not reach the required temperature due to different reasons (for instance, a cold winter day, bad isolation, broken door, and so on). The service provides `slowdown():delay`, which outputs the delay in percentage;

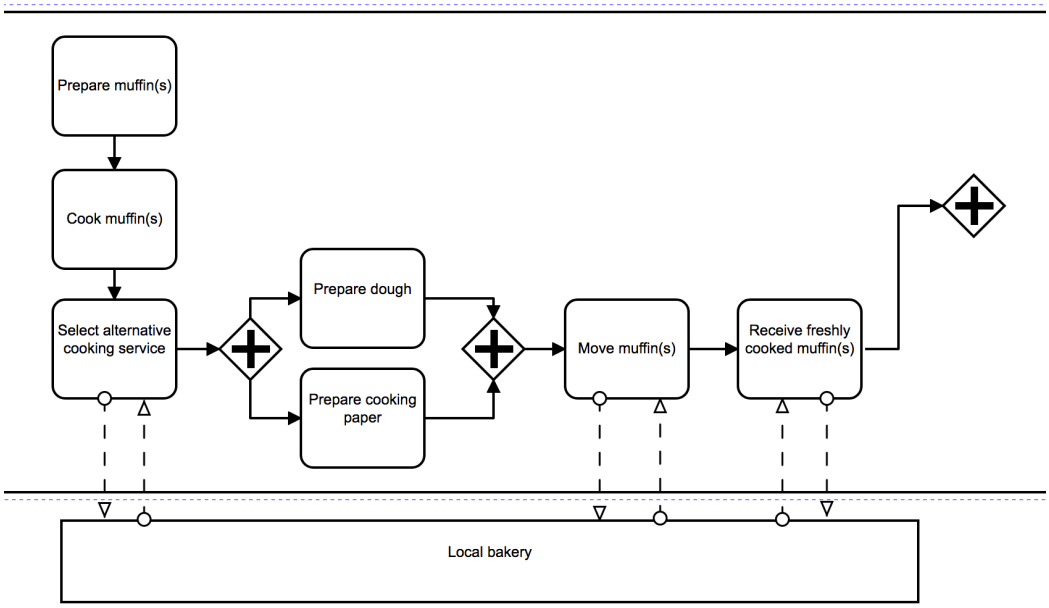

Figure 4: A fragment of the adapted process. After the original activities *Prepare muffin(s)* and *Cook muffin(s)* (cf. Figure 1), new activities are introduced, in order to *Select alternative cooking service*. Then, analogously to the original process, *Prepare dough* and *Prepare cooking paper* are performed, the muffins are moved and finally are received freshly cooked (cf. *Move muffin(s)* and *Receive freshly cooked muffin(s)*). Finally, the process prosecutes as the original one.

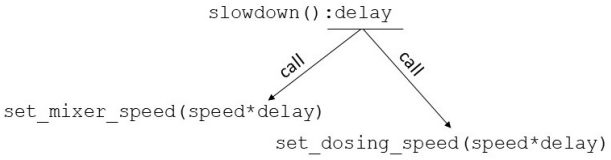

Figure 5: Service composition for adapting to oven performance.

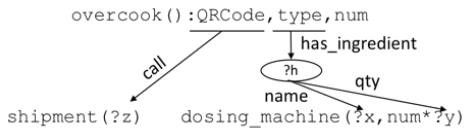

Figure 6: Service composition for the overcooked muffins.

for instance, if the oven was expected to reach the correct temperature in 30 minutes, but it actually needs 45 minutes, a delay of 33% is notified. The `slowdown()` is then composed with all the services available for reducing the speed of the machines; for instance, in Figure 5 `set_mixer_speed()` and `set_dosing_speed()` are invoked to reduce the speed of the dough mixer and of the dosing machine services.

The second example is more complex, even if related to a simple unexpected event: some muffins are overcooked, a case in which the shipping 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, `overcook():(QRCode,type,num)` is available and can be activated either by a monitoring facility or by human intervention. This service outputs the type (cf. `type`) and number (cf. `num`) of the overcooked muffins and the corresponding order (identified by its`QRCode`) and can be composed with two discovered services: one interacting with the shipping courier (e.g., `shipment(URL)` with the courier Web service as input) and one activating the dosing machine (e.g., `dosing_machine(setOfIngredients,setOfQuantities)` with ingredients and quantities as input). The composition (see Figure 6) 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 for shipment and the dosing machine.

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, or to revert to upper layers (as shown above in the case of complete process re-arrangement).

3.3. 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. Data can be static such as data available in traditional

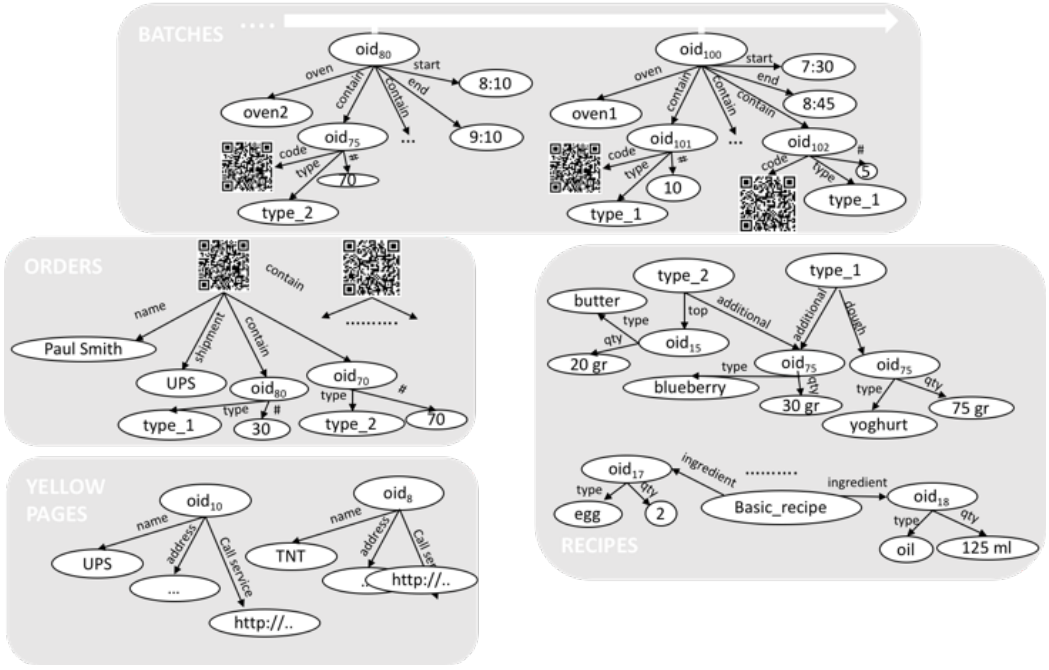

Figure 7: An excerpt of the MyMuffin data space.

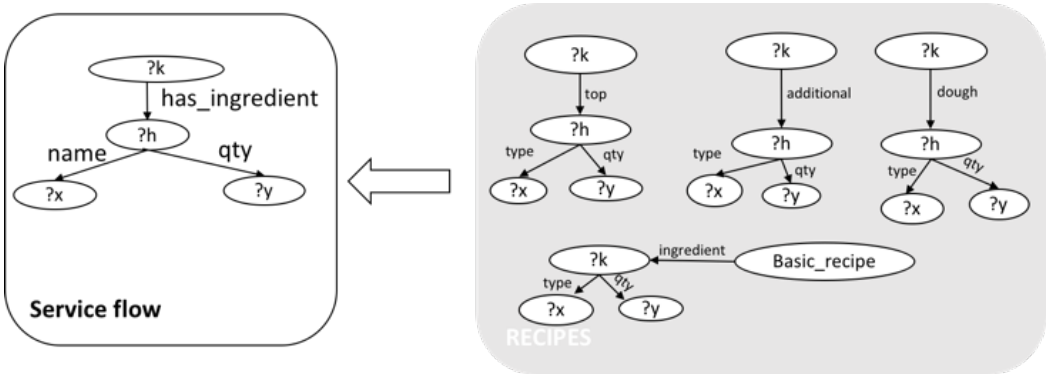

Figure 8: Mapping discovery process.

DBMSs but also highly dynamic like sensor data that are continuously generated. Moreover, the dataspace should accommodate data with different degrees of structure, from tabular to fully textual data. Finally, the dataspace 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 [37] and, on the other hand, P2P approaches for data sharing [38]. 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 (materials, machines, enterprises, value networks, and factory workers) [4]. 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 7 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 orders made by clients 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 7, like oid_{101}, are object identifiers 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 issue that 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 instances. 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.

For instance, if the user is interested in reconstructing the current status of orders, then it is necessary to merge the data stored in the data source *Batches* and the data stored in *Orders* . In this case, entity resolution is necessary because the same muffin type of the same order is represented by different *oid*s, e.g., cf. oid_{101} with oid_{80} or oid_{75} with oid_{70}); data fusion is necessary as, 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 component 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 suggests that 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 [39]. 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 quantities as input to the dosing machine.

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 8. 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 inputs to the dosing machine 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.

4. Discussion and closing remarks

In this paper, we have proposed an architectural framework for RAMI 4.0-based digital factories. The framework supports agile supply-chains through innovative technological approaches aiming at the dynamic discovery of service and data flows that best fit the requirements expressed in business process specifications and their evolution.

The proposed approach relies on a three-level architecture whose aims are to enable the interoperability among the different parts of the real factory and to ease the involvement of humans in the agile management of factory processes. Moreover, the proposed approach leverages the interactions with other actors of the supply chain, making them easier and overcoming the obstacles deriving from the possible different data formats and process management.

We now discuss factors that may call the results of the work conducted in this paper into question or diminish the meaningfulness of the results. These factors are denoted as *threats to validity*.

A first threat to validity is the possible dimension of digital factories which may adopt the proposed approach. An architectural approach as the one proposed here, may appear as more suitable to large and "traditional" manufacturing factories (e.g., mechanical, automotive, etc.) than to small factories. Indeed, as pointed out also by the EU Commission in its initiative about *digital transformation*[4], the most benefits from digital factories will be shown in small realities and in scenarios not fully automated, as the food industry. For this reason, we have presented as a case study the MyMuffin example, in order to make it evident how to successfully apply the approach in other scenarios than manufacturing.

Another threat is the lack of an extensive validation of the approach, which need to be concretely evaluated in many different situations. In order to diminish this, *(i)* we have carefully designed the approach by integrating best approaches in the different disciplines, and *(ii)* by extensively considering the existing state-of-the-art (cf. Section 2) in order to include pros and cons of each proposals, and finally *(iii)* we have presented a carefully designed and detailed case study in order to make itself an initial validation. But undoubtedly, more work is needed in order to validate the approach in several different digital factories in different business segments and activity areas.

Related to the above threats, there is the question about the repeatability of the approach. By considering, as a case study, not a traditional manufacturing scenario but a different one, as the food industry, we argue that the approach is enough general to be applied in many other scenarios, not only the ones for which it has been conceived.

Finally, the lack of a software implementation is a crucial threat to validity. On this point, we are currently working on realizing all the layers of the proposed architecture, on the basis of available research prototypes and new ones to be developed ad hoc for this research.

Our future work and next steps will mainly consist in addressing the above mentioned threats to validity, that is in the implementation of the proposed architectural framework and proof-of-concept of such an architecture, to be validated in agile supply-chain application scenarios. It means to further investigate several interesting research issues towards the implementation of a polystore with the defined characteristics, the dynamic and interactive discovery of data sources and services, the dynamic choreography of processes, services and data for supply-chain responsiveness.

Finally, we would like to remark the impact of our research, which is manifold: from the business to the technological facets. Being our approach able to make the supply chains more agile, the adoption of the proposed solution will have a concrete impact on the industrial landscape, where companies are in need of making their supply chain more agile, but often lack proper information systems able to combine the business constraints and opportunities and the ICT potential. As an example, only in Italy more than 388 000 Italian manufacturing companies are micro, small and medium size enterprises (SMEs, up to 249 employees), and they represent 99.7% of the total number of manufacturing companies and more than 60% of the total turnover (Eurostat 2018). Thus, in many cases, supply chains are not driven by big companies, which could provide a sort of stability in the relationships among the partners and foster the adoption of a common ICT infrastructure. Conversely, Italian supply chains (and this is true in many other countries) are very fragmented and dynamic, to properly satisfy the multiple different customers requesting tailored products and services. SMEs often face the challenge to interact digitally with their counterparts, lacking both standards and resources. Using a common reference architecture and an agile ICT infrastructure, our research offers a solution for these numerous enterprises by enabling the creation of a "co-opetitive" environment where companies can respond more quickly to the emerging needs of the market. The adoption of the proposed reference architecture would have also a significant impact on the Italian digital market, whose value was more than €66 billion in 2016 and growing, employing approximately 740 000 people, as companies will require to revise their information systems to make them compliant to the proposed reference architecture. Although this could be seen as a cost that the companies have to bear, on the

[4] Cf. `https://ec.europa.eu/growth/industry/policy/digital-transformation_en`.

other side, the opportunity to simplify the participation to the supply chains and the consequent benefits will definitely compensate the effort.

It is worth to mention that the need for agile supply chains is clear also at European level, as the EFFRA association identifies "agile value networks" as one of the five key priorities for the Future of Factories to deliver innovative products with a high degree of personalization. Thus, the adoption of our proposal goes into this direction, supporting the achievement of this goal on a continental scale, in particular considering that Italian firms strongly interact with European customers and suppliers. Moreover, most European countries, are similar to Italy, i.e., with very few large companies that represent a limited share of the total turnover. At European level SMEs (including micro companies) represent 99.2% of manufacturing companies (Eurostat 2018), although with some differences among countries. Germany for example has a higher share of large firms (2.1% of the total, representing 74.5% of the turnover, compared to a European average of 62.8%). For this reason, our proposal also contribute to the digital transformation of SMEs all over Europe. At the business level, the impact of our research is clear also from the customer standpoint, as facilitating the information exchange and the possibility to react to negative as well as positive changes occurring in the supply chains can provide more customized products as well as added-value services to the customer. This is very relevant nowadays, since *servitization* is more and more pursued by companies to attract an increasing number of customers that want both customized products and services in addition to the products they buy. Moreover, agility considers security flaws among the potential risks against which supply chains need to be responsive. Therefore, the adoption of the proposed solution has a fundamental value today, considering that security and privacy of data are one of the most important issues for the public.

Acknowledgements.. This work has been partly supported by the European Commission through the H2020 project FIRST – virtual Factories: Interoperation suppoRting buSiness innovaTion (grant agreement # 734599).

References

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

[2] M. P. Papazoglou, Smart connected digital factories - unleashing the power of industry 4.0 and the industrial internet, in: Proceedings of the 8th International Conference on Cloud Computing and Services Science, CLOSER 2018, Funchal, Madeira, Portugal, March 19-21, 2018., SciTePress, 2018, pp. 260–271.

[3] F. Zezulka, P. Marcon, I. Vesely, O. Sajdl, Industry 4.0 an introduction in the phenomenon, IFAC-PapersOnLine 49 (25) (2016) 8 – 12, 14th IFAC Conference on Programmable Devices and Embedded Systems PDES 2016.

[4] E. F. of the Future Research Association, Factories 4.0 and beyond, Tech. rep., EU (2016).

[5] M. Yamamoto, H. Sakamoto, Fdt/dtm framework for field device integration, in: 2008 SICE Annual Conference, 2008, pp. 925–928. doi:10.1109/SICE.2008.4654787.

[6] H. Cai, L. Da Xu, B. Xu, C. Xie, S. Qin, L. Jiang, Iot-based configurable information service platform for product lifecycle management, IEEE Transactions on Industrial Informatics 10 (2) (2014) 1558–1567.

[7] F. Tao, Y. Zuo, L. Da Xu, L. Zhang, Iot-based intelligent perception and access of manufacturing resource toward cloud manufacturing, IEEE Transactions on Industrial Informatics 10 (2) (2014) 1547–1557.

[8] Y. Lu, Industry 4.0: A survey on technologies, applications and open research issues, Journal of Industrial Information Integration 6 (2017) 1 – 10. doi:https://doi.org/10.1016/j.jii.2017.04.005.
URL http://www.sciencedirect.com/science/article/pii/S2452414X17300043

[9] M. Dumas, M. La Rosa, J. Mendling, H. A. Reijers, Fundamentals of Business Process Management, Second Edition, Springer, 2018. doi:10.1007/978-3-662-56509-4.

[10] W. M. P. van der Aalst, Process Mining - Data Science in Action, Second Edition, Springer, 2016. doi:10.1007/978-3-662-49851-4.

[11] C. Janiesch, A. Koschmider, M. Mecella, B. Weber, A. Burattin, C. Di Ciccio, A. Gal, U. Kannengiesser, F. Mannhardt, J. Mendling, A. Oberweis, M. Reichert, S. Rinderle-Ma, W. Song, J. Su, V. Torres, M. Weidlich, M. Weske, L. Zhang, The internet-of-things meets business process management: Mutual benefits and challenges, CoRR abs/1709.03628. arXiv:1709.03628.
URL http://arxiv.org/abs/1709.03628

[12] A. Garcia-Dominguez, M. Marcos, I. Medina, A comparison of bpmn 2.0 with other notations for manufacturing processes, AIP Conference Proceedings 1431 (1) (2012) 593–600. arXiv:https://aip.scitation.org/doi/pdf/10.1063/1.4707613, doi:10.1063/1.4707613.
URL https://aip.scitation.org/doi/abs/10.1063/1.4707613

[13] S. Zor, D. Schumm, F. Leymann, A proposal of bpmn extensions for the manufacturing domain, in: Proceedings of the 44th CIRP International Conference on Manufacturing Systems, 2011.

[14] A. Marrella, M. Mecella, S. Sardiña, Supporting adaptiveness of cyber-physical processes through action-based formalisms, AI Commun. 31 (1) (2018) 47–74. doi:10.3233/AIC-170748.

[15] T. Erl, Service-Oriented Architecture: Concepts, Technology, and Design, Prentice Hall Professional Technical Reference, Upper Saddle River, NJ, 2005.

[16] T. Bangemann, M. Riedl, M. Thron, C. Diedrich, Integration of classical components into industrial cyber physical systems, Proceedings of the IEEE 104 (5) (2016) 947–959. doi:10.1109/JPROC.2015.2510981.

[17] M. Wagner, D. Zbel, A. Meroth, Soda: Service-oriented architecture for runtime adaptive driver assistance systems, in: 2014 IEEE 17th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing, 2014, pp. 150–157. doi:10.1109/ISORC.2014.15.

[18] H. Bohn, A. Bobek, F. Golatowski, Sirena - service infrastructure for real-time embedded networked devices: A service oriented framework for different domains, in: International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies (ICNICONSMCL'06), 2006, pp. 43–43. doi:10.1109/ICNICONSMCL.2006.196.

[19] A. W. Colombo, S. Karnouskos, O. Kaynak, Y. Shi, S. Yin, Industrial cyberphysical systems: A backbone of the fourth industrial revolution, IEEE Industrial Electronics Magazine 11 (1) (2017) 6–16. doi:10.1109/MIE.2017.2648857.

[20] E. Tello-Leal, O. Chiotti, P. D. Villarreal, Software agent architecture for managing inter-organizational collaborations, Journal of applied research and technology 12 (3) (2014) 514–526.

[21] B. l. Zhang, Service-oriented logistics supply chain information management system, in: 2018 International Conference on Intelligent Transportation, Big Data Smart City (ICITBS), 2018, pp. 428–431. doi:10.1109/ICITBS.2018.00114.

[22] T. Sakakura, A speculation on a framework that provides highly organized services for manufacturing, in: 2015 IEEE International Conference on Automation Science and Engineering (CASE), 2015, pp. 1025–1028. doi:10.1109/CoASE.2015.7294233.

[23] G. Castelli, M. Mamei, A. Rosi, F. Zambonelli, Engineering pervasive service ecosystems: The sapere approach, ACM Trans. Auton. Adapt. Syst. 10 (1) (2015) 1:1–1:27. doi:10.1145/2700321.

URL http://doi.acm.org/10.1145/2700321
[24] W. He, L. Da Xu, Integration of distributed enterprise applications: A survey, IEEE Transactions on Industrial Informatics 10 (1) (2014) 35–42.
[25] C. Schroth, T. Janner, V. Hoyer, Strategies for cross-organizational service composition, in: 2008 International MCETECH Conference on e-Technologies (mcetech 2008), 2008, pp. 93–103. doi:10.1109/MCETECH.2008.13.
[26] B. Liu, Z. Zhang, Qos-aware service composition for cloud manufacturing based on the optimal construction of synergistic elementary service groups, The International Journal of Advanced Manufacturing Technology 88 (9) (2017) 2757–2771. doi:10.1007/s00170-016-8992-7.
URL https://doi.org/10.1007/s00170-016-8992-7
[27] M. B. Karimi, A. Isazadeh, A. M. Rahmani, Qos-aware service composition in cloud computing using data mining techniques and genetic algorithm, J. Supercomput. 73 (4) (2017) 1387–1415. doi:10.1007/s11227-016-1814-8.
URL https://doi.org/10.1007/s11227-016-1814-8
[28] M. Gunzert, Compatibility and interoperability in field device integration; a view on eddl, fdt and fdi, in: 2015 54th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE), 2015, pp. 941–946. doi:10.1109/SICE.2015.7285561.
[29] M. Hepp, J. Leukel, V. Schmitz, A quantitative analysis of ecl@ss, unspsc, eotd, and rntd content, coverage, and maintenance, in: IEEE International Conference on e-Business Engineering (ICEBE'05), 2005, pp. 572–581. doi:10.1109/ICEBE.2005.15.
[30] G. De Giacomo, D. Lembo, M. Lenzerini, R. Rosati, On reconciling data exchange, data integration, and peer data management, in: Proceedings of the Twenty-Sixth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 11-13, 2007, Beijing, China, 2007, pp. 133–142.
[31] P. G. Kolaitis, Reflections on schema mappings, data exchange, and metadata management, in: Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, Houston, TX, USA, June 10-15, 2018, 2018, pp. 107–109.
[32] M. J. Franklin, A. Y. Halevy, D. Maier, From databases to dataspaces: a new abstraction for information management, SIGMOD Record 34 (4) (2005) 27–33.
[33] P. Cudré-Mauroux, M. T. OZSU, Peer Data Management System, Springer US, Boston, MA, 2009, pp. 2055–2056.
[34] V. Gadepally, P. Chen, J. Duggan, A. J. Elmore, B. Haynes, J. Kepner, S. Madden, T. Mattson, M. Stonebraker, The bigdawg polystore system and architecture, in: 2016 IEEE High Performance Extreme Computing Conference, HPEC 2016, Waltham, MA, USA, September 13-15, 2016, 2016, pp. 1–6.
[35] J. Wang, T. Baker, M. Balazinska, D. Halperin, B. Haynes, B. Howe, D. Hutchison, S. Jain, R. Maas, P. Mehta, D. Moritz, B. Myers, J. Ortiz, D. Suciu, A. Whitaker, S. Xu, The myria big data management and analytics system and cloud services, in: CIDR 2017, 8th Biennial Conference on Innovative Data Systems Research, Chaminade, CA, USA, January 8-11, 2017, Online Proceedings, 2017.
[36] A. Marrella, M. Mecella, B. Pernici, P. Plebani, A design-time data-centric maturity model for assessing resilience in multi-party business processes, Information Systemsdoi:https://doi.org/10.1016/j.is.2018.11.002.
[37] A. Marrella, M. Mecella, S. Sardina, Intelligent process adaptation in the smartpm system, ACM Transactions on Intelligent Systems and Technology 8 (2) (2016) 1–43.
[38] W. Penzo, S. Lodi, F. Mandreoli, R. Martoglia, S. Sassatelli, Semantic peer, here are the neighbors you want!, in: Proceedings of the 11th international conference on Extending database technology: Advances in database technology, ACM, 2008, pp. 26–37.
[39] F. Mandreoli, A framework for user-driven mapping discovery in rich spaces of heterogeneous data, in: OTM Confederated International Conferences" On the Move to Meaningful Internet Systems", Springer, 2017, pp. 399–417.