An architectural approach for digital factories (Extended abstract)*

Nicola Bicocchi[1], Giacomo Cabri[1], Francesco Leotta[2], Federica Mandreoli[1], Massimo Mecella[2], and Francesco Sapio[2]

[1] Università degli Studi di Modena e Reggio Emilia, Italy
[2] Sapienza Università di Roma, Italy

Abstract. Digital factories comprise a multi-layered integration of various activities along the factories and product life-cycles. 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 expands outside the company boundaries and allows to collaborate on business processes over the whole supply chain. This extended abstract, based on a recently published paper, discusses an interoperability architecture for digital factories. It analyses the key requirements for enabling a scalable factory architecture characterized by access to services, aggregation of data, and orchestration of production processes.

Keywords: digital factories · data sharing and interoperability · service oriented architectures · dynamic supply chains

1 Introduction

Production processes are fragmented across different companies and organised in global multi-tier supply chains. This is the result of the first wave of globalisation, which was partly enabled by the diffusion of Internet-based Information and Communication Technologies (ICTs)in the early 2000s. A recent wave in technology, has led to the fourth industrial revolution - or *Industry 4.0*, whose goal is to increase opportunities for firms, including small and medium enterprises (SMEs) by being able to access global customers. However, this requires the ability to adapt to different requirements and conditions, volatile demand patterns, and fast changing technologies. Therefore, supply chains need to increase their *agility* by adapting to evolving and uncertain business contexts.

* This paper is based on Nicola Bicocchi, Giacomo Cabri, Federica Mandreoli, Massimo Mecella (2019): Dynamic digital factories for agile supply chains: An architectural approach. *Journal of Industrial Information Integration*, DOI: https://doi.org/10.1016/j.jii.2019.02.001. The work has been partly supported by the European Commission through the H2020 project FIRST – virtual Factories: Interoperation suppoRting buSiness innovaTion (grant agreement # 734599).

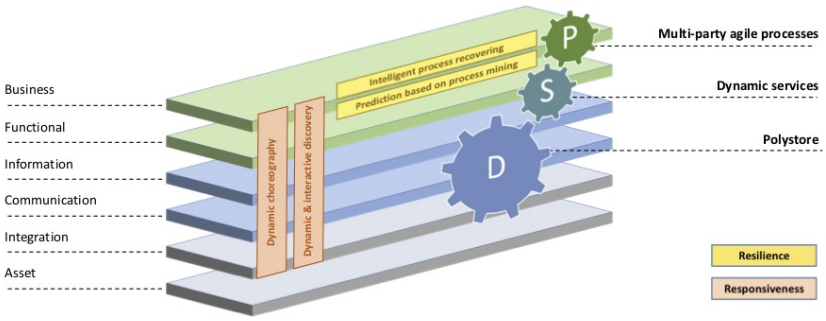


Fig. 1: Our RAMI-based architectural framework

Digital factory is a key paradigm to this end because it uses digital technologies to promote the integration of product design processes, manufacturing processes, and general collaborative business processes across factories and enterprises [2, 6]. An important aspect of this integration is to ensure *interoperability* between machines, products, processes, and services, as well as any descriptions of those. As a result, a digital factory consists of a multi-layered integration of the information related to various activities along the factory and related resources.

The main contribution of this paper is to provide methodological and technological support to agile supply chains in the Industry 4.0 context. It sets forth an architectural framework that the leverages Reference Architectural Model Industrie - RAMI 4.0 [7] and addresses the methodological issue of making RAMI 4.0 capable of enabling agility in supply chains. To address this aim, we propose an architectural framework that enables interoperability via 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. Lastly, the rest of the paper is organised as follows: Section 2 presents the RAMI 4.0-based architectural framework, and Section 3 concludes the paper and discusses future work.

2 Enabling interoperability

The approach undertaken in this work is based on RAMI 4.0. RAMI is a layered reference architectural framework in the manufacturing industry domain developed in Germany by leveraging EU initiatives and guidelines [3].

RAMI 4.0 consists of six different layers. As shown in Fig. 1, we propose a correspondence between these layers and established technologies, i.e., polystores, dynamic services and multi-party agile processes.

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

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

heterogeneity. For instance, machines equipped with IoT sensors continuously produce data streams, Online transaction processing - OLTP data are available in DBMSs, Online analytical processing - OLAP data are available in data warehouses, digital manuals are stored in repositories, and so on. To deal with these information, data sources are organised as a dataspace where they can communicate through mappings. The dataspace adhere to the polystore model supporting dynamic configurations (i.e., data sources going in and out the system).

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 to control, discover, and compose them in a dynamic way. Rich semantic descriptions of the services should be available to support both the discovery of the services and their execution/invocation. This service layer allows to achieve higher-level goals defined at the business level.

At the business level, business process specifications need to capture not only orchestrated processes but also choreographed processes across different organizations, as a supply chain definition requires. 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 [4].

In order to describe our approach we refer to the following scenario.

MyMuffin is a (fictitious) company operating within the EU producing muffins and is expanding its business to allow customers to buy muffins online (in boxes of 4). Clients can customise their muffins by choosing among different combinations of ingredients (e.g., chocolate chips), toppings (icing sugar), and additions to the dough (e.g., honey, yoghurt). The client can also customise the colors of the muffin liners (e.g., pink, yellow) as well as the colors of the box. The muffins are then delivered to a specified location.[4]

The muffin factory collects orders and organises batches of muffin doughs for production. For example, if a client orders 3 boxes of carrot muffins with yoghurt, icing sugar on top, and pink baking paper, whereas another client orders 2 boxes of carrot muffins with yoghurt, nothing on top, and yellow baking paper, the same dough can be used for both orders. Once an order is received, the muffin liners are set-up as in parallel to the dough preparation. In addition, a QR-code is printed on the baking paper to serve as a unique identifier for a specific order. After the dough has been prepared, the muffins are placed in muffin liners and sent to the oven (connected to a QR code reader) for cooking. Muffins are cooked in batches of 1000 muffins. Once the muffins are cooked, toppings are placed, the cart is operated to route the different muffins to the right boxes where they are then ready for delivery. Considering the steps involved here, agility is needed during each stage of process, (e.g., the baking step may overcook some muffins, which therefore are not ready for delivery and should be prepared again).

[4] MyMuffin is a fantasy company, but there are real successful examples of mass customization applied to food likewise Mymuesli, a German company - `https://en.wikipedia.org/wiki/Mymuesli`.

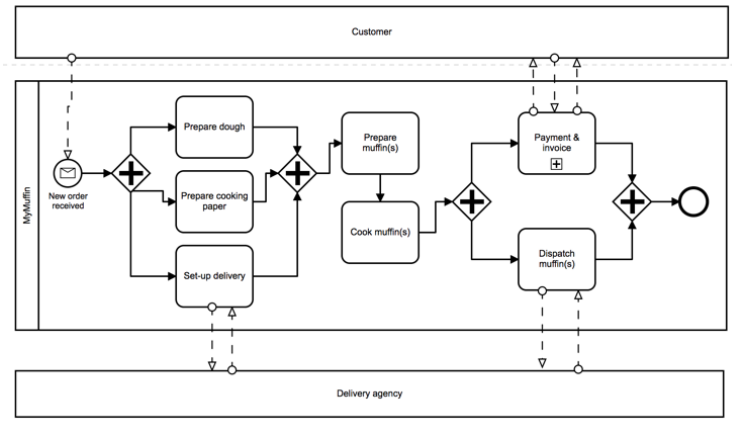


Fig. 2: The BPMN diagram describing the MyMuffin process.

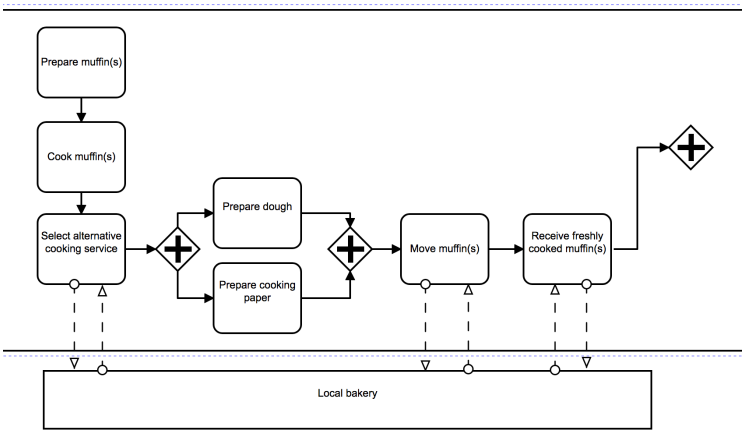


Fig. 3: A fragment of the adapted process.

2.1 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. Figure 2 shows the process behind MyMuffin represented using Business Process Model and Notation (BPMN) [5].

In the MyMuffin example, some goals of the process may be:
[G1] for each order, evade it within 36 hours;
[G2] for each order, the final delivery to the customer should be within 72 hours.

[5] See `http://www.bpmn.org/`

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., 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) that are defined over many aspects (e.g., interactions with external companies being part of the process, maintain a total of less than 24 hours from pick-up to delivery of orders, and to keep a KPI of 95% respected over the week). 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. Through automated planning techniques, like the one adopted in SmartPM [5], the process can be modified. In particular, as shown in Figure 3, after the original activities *Prepare muffin(s)* and *Cook muffin(s)*, new activities are introduced, 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 (see *Move muffin(s)* and *Receive freshly cooked muffin(s)* tasks). Finally, the process continues as the original one. Notably, this is only one of the possible adaptations.

2.2 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 to support both discovery and service execution. The descriptions should include keywords identifying the context of the service (e.g., "food", "cooking"), the equipment (e.g., "oven", "mixer"), the operation (e.g., "turn-on", "speedup"), and the parameters (e.g., "temperature", "speed").

In regards 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 (such as the adaptation techniques realised in the process layer fail, and a human intervention is needed 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 description of the service parameters is needed to exploit the functionalities of the data layer to adapt the client service invocation to the server syntax. Some proposals and examples of semantic service descriptions are proposed in [1].

As an example, consider a scenario where the oven does not reach the required temperature due to different reasons (e.g., a cold winter day, bad isolation, broken door). The oven service provides the `slowdown():delay` operation, which outputs the delay in percentage (see Fig. 4a); for instance, if the oven was expected to reach the correct temperature in 30 minutes, but it actually needs 45

(a) Adapting to oven performance (b) Adapting to overcooked muffins

Fig. 4: Service composition.

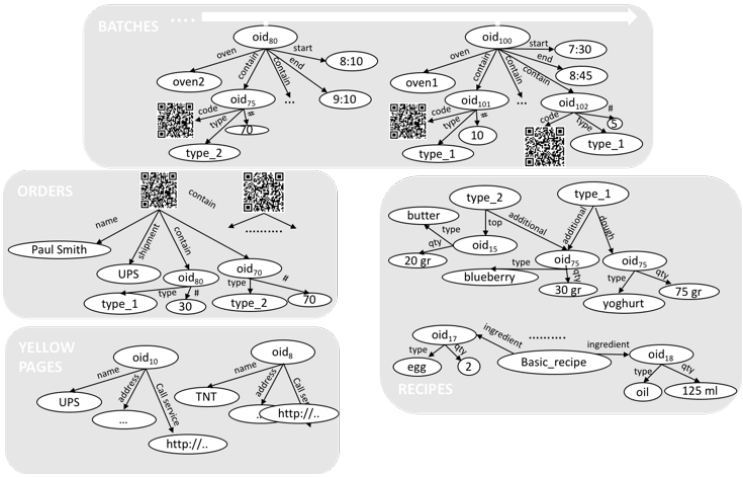
minutes, a delay of 33% is returned. The operation is then composed with all the services available for reducing the speed of the machines.

As a second example, consider the case where some muffins are overcooked (see Fig. 4b). In this case, the shipping courier must be notified to modify the shipment, and a new set of muffins must be produced starting from the list of ingredients. To this aim, the `overcook():(QRCode,type,num)` overation is available and can be activated either by a monitoring facilities or by human intervention. This operation outputs the type `type` and number `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 (i.e., `shipment(URL)` with the courier service as input) and one activating the dosing machine (i.e., `dosing_machine(setOfIngredients,setOfQuantities)` with ingredients and quantities as 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.

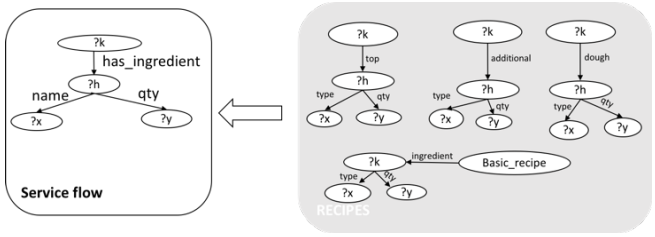
2.3 Data space layer - service-oriented mapping discovery

Data are managed and accessed in a data space. The data space must be able to deal with a huge volume of heterogeneous data from 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 data space level. Data can be static such as those available in traditional DBMSs but also highly dynamic like sensor data. Moreover, the data space should accommodate data with different degrees of structure, from tabular to fully textual data. Finally, the data space 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 space is a collection of heterogeneous data sources that can be involved in the processes, both in-factory and out-factory, and that can exchange data through mappings, i.e. declarative specifications describing the relationship between a target data instance and possibly more than one source data instances. 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, Fig. 5a shows a small portion of the MyMuffin data space that can be used in case of overcooking where data are modelled as triples. *Batches* is a data stream that reports the cooking status over time; *Orders* is the



(a) An excerpt of the MyMuffin data space.



(b) Mapping discovery process

Fig. 5: The data space.

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.

In this large scale deeply heterogeneous and dynamic integration scenario, unlike traditional approaches, mappings are interactively created and refined according to the needs of service flows and the exclusive role of mappings is to contribute to execute service compositions [3]. Hence, we start from a chain of services with their information needs expressed as inputs and outputs that we attempt to satisfy in the dataspace. For instance the data flow of Fig. 4b 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

Fig. 5b. 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.

3 Conclusion

In this paper, we have outlined 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.

Our future work and next steps will mainly consist 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.

Finally, we would like to remark the impact of our research. At the business level, the potential impact of our research can facilitate the information exchange and ability for companies to increase their agility by adapting to changes occurring in the supply chains. As a result, they can offer more customised products and services to the customer. 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 as security and privacy of data are one of the most important issues for the public.

References

1. G. Castelli, M. Mamei, A. Rosi, and F. Zambonelli. Engineering pervasive service ecosystems: The sapere approach. *ACM Trans. Auton. Adapt. Syst.*, 10(1).
2. N. Chungoora, R. I. Young, G. Gunendran, C. Palmer, Z. Usman, N. A. Anjum, A.-F. Cutting-Decelle, J. A. Harding, and K. Case. A model-driven ontology approach for manufacturing system interoperability and knowledge sharing. *Computers in Industry*, 64(4):392–401, 2013.
3. F. Mandreoli. A framework for user-driven mapping discovery in rich spaces of heterogeneous data. In *OTM Conf.*, pages 399–417, 2017.
4. A. Marrella, M. Mecella, B. Pernici, and P. Plebani. A design-time data-centric maturity model for assessing resilience in multi-party business processes. *Information Systems*, 2018.
5. A. Marrella, M. Mecella, and S. Sardiña. Supporting adaptiveness of cyber-physical processes through action-based formalisms. *AI Commun.*, 31(1):47–74, 2018.
6. M. P. Papazoglou. Smart connected digital factories - unleashing the power of industry 4.0 and the industrial internet. In *Proc. of CLOSER*, pages 260–271, 2018.
7. F. Zezulka, P. Marcon, I. Vesely, and O. Sajdl. Industry 4.0–an introduction in the phenomenon. *IFAC-PapersOnLine*, 49(25):8–12, 2016.