Architectural Concerns for Digital Twin of the Organization

Chapter · September 2020

DOI: 10.1007/978-3-030-58923-3_18

6 authors, including:

Mauro Caporuscio
Linnaeus University
75 PUBLICATIONS **989** CITATIONS
SEE PROFILE

Farid Edrisi
Linnaeus University
7 PUBLICATIONS **34** CITATIONS
SEE PROFILE

Diego Perez-Palacin
University of Zaragoza
44 PUBLICATIONS **409** CITATIONS
SEE PROFILE

Architectural Concerns for Digital Twin of the Organization

Mauro Caporuscio[1], Farid Edrisi[1], Margrethe Hallberg[2], Anton Johannesson[3], Claudia Kopf[2], and Diego Perez-Palacin[1]

[1] Linnaeus University, Växjö, Sweden
{mauro.caporuscio, farid.edrisi, diego.perez}@lnu.se
[2] Scania AB, Oskarshamn, Sweden
{grethe.hallberg, claudia.kopf}@scania.se
[3] Virtual Manufacturing AB, Göteborg, Sweden.
anton.johannessone@virtual.se

Abstract. Employing a Digital Twin of the Organization would help enterprises to change and innovate, thus enhancing their organization's sustainability. However, the lack of engineering best practices for developing and operating a Digital Twin of the Organization makes it difficult for enterprises to fully benefit from it. Many companies are currently investigating the potential use of it, but available solutions are often context-dependent or system-specific, and challenging to adapt, extend, and reuse. Therefore, digitalization is perceived as a slow, resource-demanding, and extremely expensive process whose outcome is uncertain. To this extent, enterprises seek solutions allowing them to gently introduce a Digital Twin of the Organization into their organization and to evolve it according to the changing needs and situations. This paper reports a first attempt on architecting a Digital Twin of an Organization, and discusses some architectural concerns to be addressed in order to facilitate its development and evolution.

1 Introduction

The digitalization of industry is a grand challenge. The advent of Cyber-Physical Systems (CPS) combined with Data-Driven (DD) technologies creates the necessary infrastructure for the 4th industrial revolution, where machines and humans are interconnected in sociotechnical systems, which are business or mission-critical [5]. Indeed, most sociotechnical systems are *organizational systems* which support enterprises to achieve their goals by explicitly including all those artifacts, processes and people needed to run the business. In this context, having an accurate digital representation of the organization would facilitate the decision-making processes in the value chain by providing (*i*) an aggregated view of the elements affecting a decision, and (*ii*) a prediction of the decision outcome: namely, a *Digital Twin of the Organization* (DTO) [11].

A Digital Twin of the Organization allows for representing all the elements and connections of a organizational system in virtual models, which can be

perpetually simulated and analyzed to achieve *continuous assessment* and *optimization* of the organization. Employing a well-defined DTO is an important asset in positioning new developments within the context of existing processes and other assets of an organization, as it helps in identifying necessary and/or opportunistic changes. Indeed, good DTO practices can help organizations to change and innovate, thus enhancing the organizations *sustainability*.

However, the digitalization process is far from being fast, inexpensive, or effortless. The lack of engineering best practices for developing and operating DTOs makes it difficult for industry to fully benefit from them. Many companies are currently investigating the potential use of DTOs, but provided solutions are often context-dependent or system-specific, and challenging to adapt, extend and reuse. Indeed, making optimal business/design decisions and selecting the right technology (given the considerable number of DTO, CPS, and DD technologies and platforms available) able to satisfy both current and future needs is challenging. The following main question, raised by industry, should be addressed:

IQ: "*Digitalization can't happen overnight, but should be a long step-by-step journey into the future from where we start today. How to gently introduce a DTO satisfying current needs, and then evolve it according to changing situations?*"

In fact, the adoption of DTO is, as we describe above, not straightforward and the many challenges prevent organizations from deciding to move into and invest in DTOs and related technology. Some decisions, among others, are: (1) How to incorporate CPS, DD and DTO technologies into an existing organization? (2) How to benefit from incorporating DTO in products and/or production? (3) How to define and achieve a long-term strategy for adopting DTO? (4) How to access and analyze data in real-time and enact changes based on decision support mechanisms? (5) When the organization executes structural changes, how to systematically evolve the DTO to reflect these changes?

To this extent, engineering best practices suggest the exploitation of architectures (at different levels of abstraction) to capture different aspects of the system under development. Indeed, separation of concerns (e.g., structure from behavior) improves the design and eases the maintenance of the system.

In this paper, we report about a first attempt made in architecting and developing a DTO, and evolving it to accommodate changes occurred at the business level. Evolvability is investigated through development. Specifically, we first investigate how to employ a DTO in an already-in-place manufacturing process (representing only a small part of the whole production system), then we evolve it by developing and integrating, new features. The paper details the development process, and pinpoints a set of concerns to be confronted with when architecting a DTO that is expected to evolve overtime according to the organizations requirements. From the experience, it emerges the need for defining an architectural framework for facilitating DTO development and evolution.

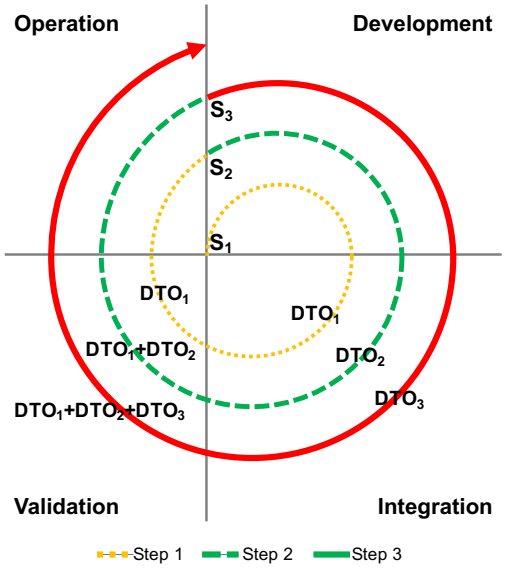


Fig. 1: Incremental approach

The paper is organized as follows. Section 2 reports our experience in implementing and evolving a DTO for a given industrial case study, whereas Section 3 pinpoints and discusses some architectural concerns to be addressed in order to facilitate the DTO evolution. Section 4 discusses related work, and Section 5 conclude the paper by addressing future research directions.

2 Industrial Case Study

This Section describes the first steps towards the digitalization of an existing production line. It provides details on the DTO development, and some examples about how the organization can benefit from the DTO.

2.1 Digitalization journey

The digitalization journey in the organization requires the construction of the DTO focusing on its evolvability. *Evolvability* is a design principle that takes into account the future change and growth of the system. Within the incremental digitalization process, it serves as a measure of the ability to extend and modify DTO upon the introduction of new activities to be digitalized or new requirements to be fulfilled. We use the evolvability measure to assess the level of effort necessary to evolve the DTO and put it into operation.

As a key requirement for a DTO is to mimic the structure of the organization that produces it, we leverage on *Enterprise Architecture* [20] (EA) to describe

the different aspects of a DTO. In fact, EA is by default positioned to play a key role in realizing a DTO, as it embeds all the *principles* and *models* used in the realization of an organization: (*i*) the *Organizational Architecture* (OA) and *Business architecture* (BA) provide the structural and behavioral models, respectively, whereas (*ii*) the *Information Architecture* (IA) provides the data representing the actual status of the organization.

We adopt the incremental development approach in Figure 1, where each evolution step is divided into four phases: *development*, *integration*, *validation*, and *operation*. As illustrative example, consider that we start from a subsystem S_1 that has already been digitalized into DTO_1. We incrementally digitalize the organization by taking another subsystem S_2 and developing its DTO_2, implementing the necessary integration of digitalized subsystems and validating it, resulting in $DTO_1 + DTO_2$.

2.2 Organization description

The organization is a supplier of articles in the automotive industry. It manufactures a product line with a large number of variants (≥ 1000). The automotive industry supply chain requires Just-In-Time production. This requirement prevents from, for instance, producing to a buffer. We focus only on one of the different activities, namely the manufacturing of a specific part P of the final product. To keep the appropriate synchronization among all parts of a product in the whole assembly line, there is an additional requirement for P: units must leave their production line in the same sequence as they were ordered.

Figure 2 illustrates the OA/BA representing the production line P according to the notation introduced in [20], which consists of two workstations: Workstation A (WA) manufactures the product and it is highly automated with two machines working in parallel (WA1 and WA2). An operator must supervise the machines and must change their configuration between orders of a different variant of P. Workstation D (WD) packages the product. It also has two machines working in parallel (WD1 and WD2), but it is less automatized and requires two operators continuously working on it, one on each machine. The business process is supported by a legacy in-house software for Enterprise Resource Planning (ERP), which constitutes the IA. Data about both the products and the production activities is collected by the operators, which are required to check the products (e.g., for identifying defects), to observe the process events (e.g., for identifying deviations in the production plan), and to report all their observations into the IA.

This section of the plant receives orders to produce P. Each order refers to a concrete variant v of P. An order specifies the number n of units to produce. The operator in WA will configure either WA1 or WA2 to produce n products P of variant v. When an order is received and both WA1 and WA2 are busy, it waits for its manufacturing turn in a queue. When the manufacturing in WA completes, P moves towards its packaging in WD. Operators in WD create packages containing up to 30 units. If both operators in WD are busy when P arrives for its packaging, then P joins a priority queue. The queue uses as

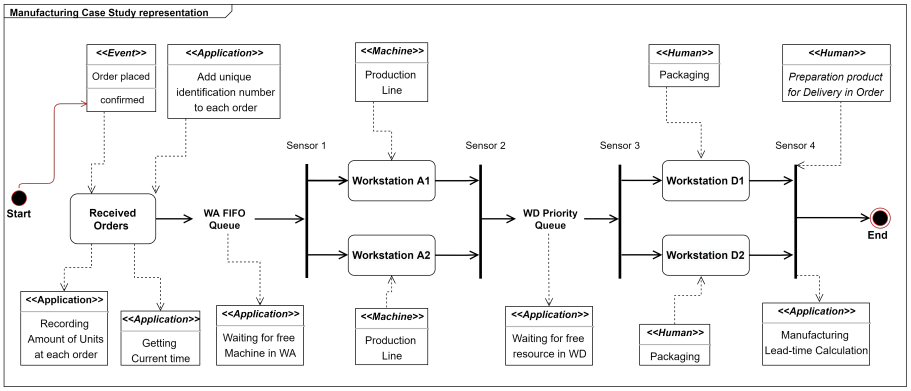


Fig. 2: OA/BA of the case study

priority the sequence number of the order. Older orders have more priority than newer orders, to help preserve the in-order delivery requirement.

The organization stakeholders (e.g., management and operators) want to digitalize this section of the plant. The reason is that they would like to know, in real time, the orders lead time, the utilization of resources, and the amount of work-in-progress in the line. They would also like to predict the near future of these three characteristics. These predictions will allow the sales department to give better estimations of the delivery time to the clients at the moment they are filling an order, and the production planning department can plan less disruptive maintenance of machinery.

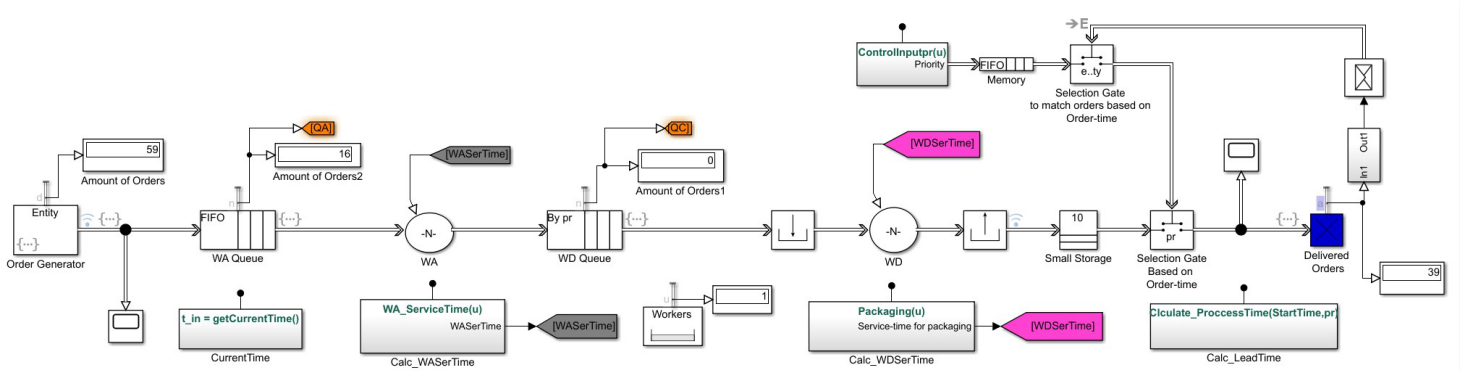
2.3 DTO Development – Iteration 1

According to the IA, the monitorable data in the production line are: According to the IA, the monitorable data in the production line are:

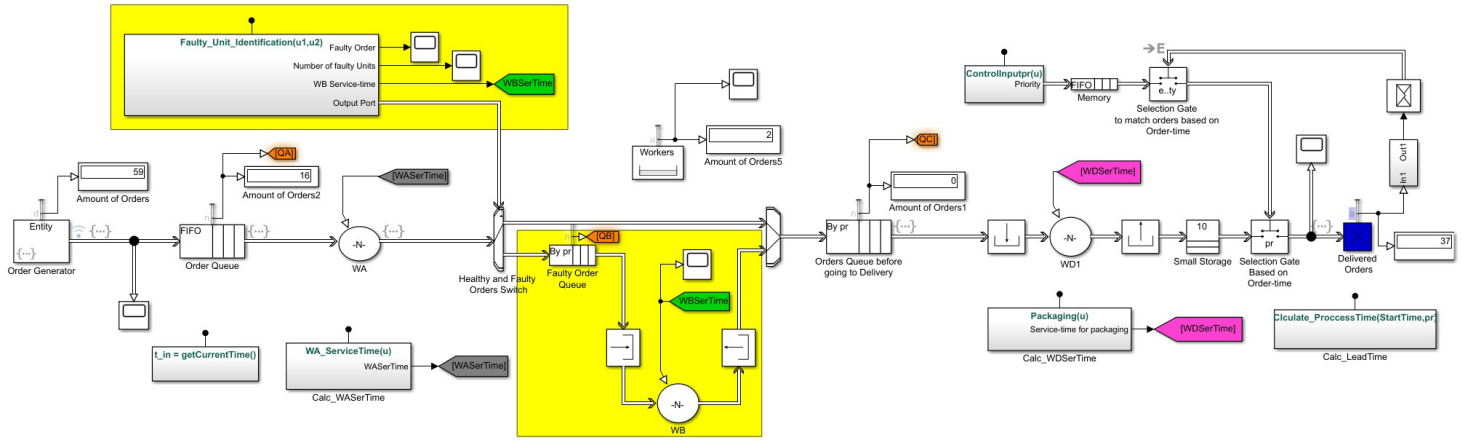
- the moment when an order is placed, and its unique identifier ID;
- the type of variant v, and the number of units, n, to produce;
- the moment when an order ID enters WA;
- the moment when the units belonging to a given order ID are produced;
- the moment when the units enter WD;
- the moment when an order ID is packaged and ready to be delivered; and,
- the moment when an order ID is delivered.

Figure 3a shows the digitalization of the production line P based on the monitorable points. The model, developed using the Discrete Event System toolbox in Matlab/Simulink[4], shows the sequence of order arrival, waiting in queue for WA, execution in WA, waiting in queue for WD, execution in WD, and in order delivery. The last is modeled with a selection gate that controls the orders ID

[4] https://se.mathworks.com/solutions/discrete-event-simulation.html

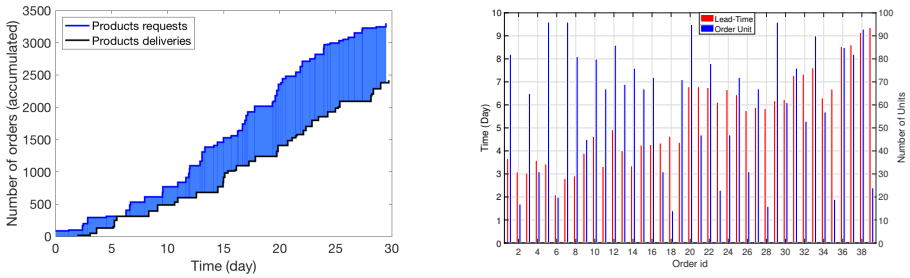


(a) Digital Twin of the Organization in the first iteration



(b) Evolved Digital Twin of the Organization

Fig. 3: Digital Twins of the Organization (model for simulation)



(a) Orders size, arrival and delivery times (b) Orders lead time

Fig. 4: Observations from DTO

and a small storage. The small storage stocks some orders if necessary until the order with the next ID to deliver is packaged in WD.

This DTO allows for observing the status of the plant –in terms of lead time of orders, utilization of workstations, amount of work-in-progress, products in queue waiting for an an available workstation–, and simulating future scenarios.

Figure 4 shows this type of observations. The horizontal axis indicates the observation time, 30 days. Figure 4a shows the arrival (blue line) and delivery (black line) of orders along time, and the size of each order. In the last 30 days, 59 orders arrived to the production line, adding up 3303 units to produce. The blue area that is demarcated by these two lines exposes the work-in-progress in the line. The height of the blue area in each x value exposes the work-in-progress in the line in that moment.

Figure 4b illustrates the lead time of each order. The number of units to produce in a single order is variable in the interval [1,100]. Since the size of an order affects its lead time, Figure 4b puts together a bar for the lead time of an order and a bar for the number of units n in the order.

2.4 DTO Development – Iteration 2 (evolution)

The organization would like to predict more accurately the lead time and to estimate in real time what is the optimal task for an operator. To achieve these new goals, the organization needs to evolve its DTO.

The DTO evolution consists in providing a more realistic representation of what happens in the organization between the moment an unit leaves machines WA1 or WA2 and an order is ready for packaging. In the real production line, machines in WA do not always manufacture the units correctly. Sometimes they have defects and they need a subsequent rework before joining the queue for packaging. The rework is manually done, and requires one of the operators in WD. We refer to the new manual rework station as WB.

This evolved DTO will enable a more accurate estimation of the lead time of an order –coming from the explicit digitalization of WB– and a better under-

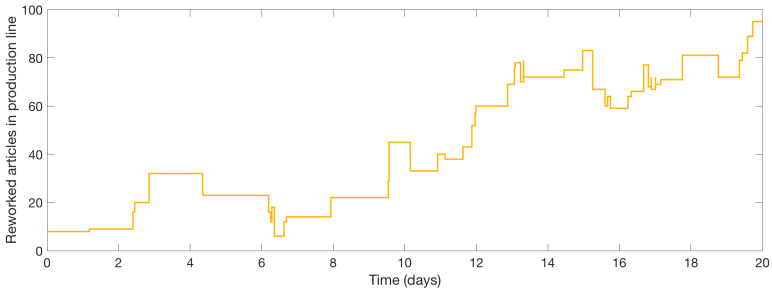


Fig. 5: Number of units in the production line that need rework

standing of the utilization of resources –coming from the more accurate representation of operators' tasks. In the first iteration in Section 2.3, the busy time of stations WD1 and WD2 also included the time that operators spent in the rework station. The separation of the operators' tasks into their actual work in WD and their work in WB enables a more accurate understanding of the busy time and service time of packaging activities in WD. Using this improved understanding, the DTO can assist the production planning in scheduling the optimal task for an operator in a given moment.

To gain this new information, it is necessary to first obtain new monitored data. This is achieved by connecting the quality sensors in the production line that inspect units for possible defects to their digitalized representation in the DTO. The operators gain awareness of defects at a glance since each quality sensor, upon the detection of a defect, sends a notification to a control panel and raises a short audible alarm. The DTO development in this phase includes that the notifications of each sensor are also sent to the DTO (the alarm is not). In case of defective unit, the DTO receives precise information of the necessary rework and, therefore, the expected lead time of its associated order is recalculated. This dynamic recalculation increases the accuracy of the lead time estimation. As soon as the quality sensors discover that an order will need a time-consuming "extra" activity (the number of defects from WA is relatively low), the information is sent to the DTO and integrated into the lead-time prediction calculations.

According to the incremental approach in Figure 1, in this iteration we first need to develop the DTO of WB. After that, we need to integrate it with the previous DTO in Figure 3a, considering the new data acquisition points and the sharing of human operators among WD and WB. Figure 3b illustrates the resulting model of the integration, where highlighted (yellow) areas represent the new parts that have been added to Figure 3a to evolve the initial DTO.

The evolved DTO can be analyzed to estimate the most efficient allocation of operations in each moment, and to easily control the evolution of the number of units that require manual rework (Figure 5 illustrates an example of this information) and the number of operators in WD and WB.

From the description in previous paragraphs, one can deem the DTO incremental evolution as a simple activity. However, it is not. The next section describes the difficulties and complications arose during the DTO integration and validation phases at iteration 2, and the lessons learned during the journey.

3 Architectural Concerns

As discussed in previous section, evolving the production line and the DTO accordingly is essential for the sustainability of the organization. To this end, we reported about the physical changes made in the production line (e.g., a new workstation WB, and new sensors in WA), as well as the changes made in the initial DTO to reflect the changes in the production line.

3.1 Architectural concerns in the case study

By comparing the two DTOs depicted in Figure 3a and Figure 3b, it is clear that changes mainly occurred between WA and WD. We introduced in WA a set of different sensors, which are the quality sensors that identify defective units. As these quality sensors send different type of raw data (e.g., ASCII, Binary, Hex, and etc.) at different points in time, we added in the DTO a pre-processing entity (i.e., the `Faulty_Unit_Identification` in Figure 3b), which is in charge of analyzing sensors data, tag defective articles, and aggregate data by number of defective articles in a concrete order. It is worth noticing that, due to the volume of raw data, this activity is complex and time consuming. After the analysis, the output of `Faulty_Unit_Identification` consists in: (*i*) the order ID, (*ii*) the number of defective articles in such an order (can be zero if none of the articles had any defect), and (*iii*) the predicted necessary time for reworking the defective articles (can be zero). Using this information, the DTO recalculates the estimation of the manufacturing lead time for the work-in-progress orders.

An order proceeds its way to either WD or WB according to the existence of defective articles. If the order is healthy (i.e., the number of defective units is 0) it is transferred to the priority queue that precedes WD, otherwise it is transferred to WB. When entering WB, the order is buffered into a FIFO queue and waits for an operator. Once the rework is finished the order proceeds to the priority queue before WD, while the operator can remain working in WB or go to WD. The DTO records the defective order ID, the number of defective units, and the actual rework time. The DTO uses this information to increase the accuracy of the future estimations of the necessary rework time and, in consequence, to increase its future lead time predictions accuracy.

Further, the evolved DTO also incorporates a policy that computes how to optimally allocate operators to WB and WD to minimize lead time. It uses the information about the status of the production line and the number of articles in each stage, including articles under rework and waiting for rework. In fact, while DTO_1 (see Figure 3a) was not aligned with OA/BA because it naively assumed

two human resources continuously working in WD, the evolved DTO can also suggest the calculated optimal allocation to the operators in each moment.

For example, suppose that the two operators are working in WD. When the quality sensors detect a defective article, one of the workers in WD may decide to attend the defective article and rework it. However, this decision can result in a sub-optimal allocation of resources. It is possible that the optimal allocation still assigns two operators to WD, maybe because the current bottleneck in the production line is still WD. The evolved DTO can continuously provide a suggestion about the optimal allocation operators.

From this discussion, it is clear that making an evolved DTO is challenging, as it requires to consider many different dimensions. Summarizing, the main issues emerge from:

- The OA, BA, and IA are misaligned, and it is difficult to understand the relationships between different resources, between resources and processes, and between resources, processes and the available data.
- Both OA, BA and IA are not flexible and they are hard to evolve. When introducing/changing resource/processes/data all of the architectures need be rearranged by introducing technical debts, which will lead to architecture erosion.
- DTO is not flexible and hard to evolve. When changes happen into either the OA or the BA or the IA, the whole DTO requires to be rearranged by still introducing technical debts, which will lead to architecture erosion.

3.2 Lessons learned for sustainable digitalization

From the above discussion it emerges that, in order to facilitate a straightforward and sustainable digitalization of the structure, products, operations, and all relations tying these together, the OA/BA and IA must be fully *aligned*. In fact, once the OA/BA and IA are aligned, they can be straightforwardly mapped to the DTO and used for simulating and analyzing the organization: (*i*) OA/BA provides all the structural and behavioral models needed for the simulation and analysis of the organization, whereas (*ii*) IA provides both raw (from CPS) and aggregated (from DD) data representing the actual status of the organization.

Unfortunately, this is a necessary but not sufficient condition. In fact, as the sustainability of a DTO is largely determined by the sustainability of OA/BA and IA, it must be flexible and able to accommodate continuous and rapid change to OA/BA and IA. Indeed, if the OA/BA requires to be changed in response to new market forces (e.g., manufacturing a new product), the IA should change as well and fulfill the new requirements (e.g., a functionality for managing the new production process). Therefore, the DTO should *evolve* accordingly and reflect the changes made in both the OA/BA and IA. In fact, if the DTO does not evolve and align according to the enterprise, its benefits are drastically reduced. In other words, *rigid architectures restrain the sustainability and evolution of the DTO, and reduce its benefits.*

To this end, we need to define Architectural Frameworks, which provide principles and practices for jointly specifying, aligning, and evolving OA/BA, IA and DTO. In particular, the Architectural Framework shall include [4]: (1) an *Architectural Pattern* describing architectural elements and connectors, with constraints on how they can be combined, achieving flexibility; (2) a *Reference Model* decomposing functionality into elements (and data between them) that cooperatively satisfy the organization needs; (3) a *Reference Architecture* mapping the Reference Model onto system elements and defining the data flows between them. Whereas a reference model splits functionalities, a reference architecture is the mapping of such functionalities onto a system decomposition. The Architectural Framework should explicitly address the following concerns:

Modularity – Software engineering defines abstraction as the process of identifying the important aspects of a phenomenon and ignoring the details that are not relevant. What must be considered as important and what should be considered as a detail to be ignored depends on the purpose of the abstraction. Modularity is the degree to which DTOs are decomposed and recombined in terms of abstractions. Specifically, a modular DTO should be structured into identifiable entities, which encapsulate the representation of abstractions, and are accessible through well-defined interfaces. A modular DTO can be then composed entirely from a set of constituent DTOs, where each of them takes care of a specific aspect (i.e., *single responsibility*), rather than having the aspect spread out over many parts of the DTO. Modularity and Single Responsibility are then important concerns, as they allow for separation of concerns, which can be applied in two phases: first, when dealing with each constituent DTO in isolation (bottom-up design) and second, when dealing with the overall characteristics of all constituent DTOs and their relationships as a whole DTO (top-down design).

Granularity – Granularity defines the level of detail of a single constituent DTO. Indeed, it refers to the extent to how small the constituent DTOs are. When a DTO is split into constituents, it is important to measure the degree of componentization. On the one hand, having small, fine-grained constituents provides greater flexibility when composing them into a DTO. On the other hand, small, fine-grained constituents are more difficult to govern. In fact, the larger and coarse-grained constituents are, the easier it is to manage and coordinate them in a whole DTO. Modularity and granularity are tightly coupled and affect each other. A DTO can have multiple possible levels of granularity. The granularity of the data that the physical system can transmit and that can feed the DTO is also a factor to consider during the decision of the most appropriate DTO granularity.

Decomposition/Composition Style – Understanding how to properly decompose/compose a DTO into/from constituents is relevant for defining the modularity and granularity of the different entities. A new DTO should be built by taking reusable constituents and combining them to form the required functionality. The capability of understanding each constituent DTO in isolation

aids the adaptation and evolution of the whole DTO. To achieve modular decomposability/composability, and understanding, constituent DTOs must have two very important properties, namely *high cohesion* and *loose coupling*. A constituent has high cohesion if all its elements are strongly related, and they are together for a reasonable and well-defined specific purpose (i.e. single responsibility). Complementary, coupling is determined by the relationship of an individual constituent with other constituents in the same DTO. Coupling is a measure of the interdependence of two or more constituents in the form of interaction. If two constituents depend on each other heavily - i.e., they are tightly coupled - it will be difficult to observe, analyze, understand, and act them separately. In contrast, modular structures with high cohesion and low coupling allow for considering constituent DTOs as black boxes when the overall structure is described. Such constituents can be observed, analyzed and described separately.

Evolution – A DTO emerges from the interactions among the constituent DTOs. Despite the adoption of best engineering practices, the DTO must face inherently continuous evolution, as well as heterogeneous and inconsistent changing elements. Changes inside an organization are the cause of continuous DTO evolution. A DTO evolution occurs when any of the following three basic situations happen [8]: *self-evolution*, *joint evolution*, and *emergent evolution*.

In *self-evolution*, a change is introduced because of redesign, redevelopment or improvement of an existing DTO. Requirements include improved business-supporting functions, improved performance through using a new design and more advanced technologies, improved architecture for future development, and their combinations. Referring to the case study addressed in Section 2, an example could be an improvable production line WA able to accommodate newly designed robotic arms that best fit the product the machine is working on, or arms that are simply offering improved performance over the previous ones. Self-evolution targets a certain level of granularity of a DTO. In fact, self-evolution can occur at the highest level of granularity, or at lower levels. As the granularity of a DTO is strictly tied with its modularity, it is important to have a distinct vision of the constituents and the grains of a DTO to better locate where the evolution occurs.

In *joint evolution*, two or more DTOs are to be integrated for improved business support. Still referring to the case study in Section 2: the two communicating constituents, namely WA and WD, have to deal with possible multiple changes involving, interoperability, data sharing, improved functions and services, and workflow integration. Joint evolution targets two or more DTOs at the same or different levels of granularity. Furthermore, integration between DTOs could potentially change the connectors of the constituents involved.

Emergent evolution deals with the design and development of a new DTO based on existing DTOs. Requirements include new functions developed on a joint basis of existing constituents, and new DTOs supporting emerging business needs. An example could be the development of a DTO on the basis of the two constituents including WA and WD. The new DTO digitalizes the functionalities

of the underlying smart manufacturing system. Emergent evolution targets two or more DTOs where the one emerging has a higher level of granularity.

The previously described evolution types could result in isolated changes. However, *Multiple evolutions* could also come out simultaneously, with self-evolution, joint evolution and emergent evolution directly affecting a large part of the DTO. If the evolutions complexity is not controlled the DTO will erode over time due to such changes. The laws of software evolution [13] state that uncontrolled evolution brings to

- *Increasing complexity:* as a large program is continuously changed, its complexity, which reflects deteriorating structure, increases unless work is done to maintain or reduce it.
- *Continuing grow:* the functional capability of E-type systems must be continually enhanced to maintain user satisfaction over system lifetime.
- *Declining quality:* unless rigorously adapted and evolved to take into account changes in the operational environment, the quality of an E-type system will appear to be declining.

Hence, an open and connected DTO requires that evolution dimensions are controlled to prevent costly reconstruction. Moreover, it is important to understand **why**, **how**, **when** and **where** this evolution occurs.

4 Related Work

Related work is manifold and spans over different topics, namely *Digital Twin Architectures*, *Architecture Alignment* and *Architecture Evolution*.

Digital Twin (DT) Architectures – Since the DT conception, more than a decade ago [21], pioneers have used it in the manufacturing domain in order to improve the production leveraging *Virtual Factory* [12]. In the Industry 4.0 domain, the simulation of different possible behaviors is a main enabler for construction of the desired intelligent systems [19]. Despite the efforts to systematize and guide the development and deployment of digital twins in companies [1, 9], the systematic implementation of digitalization process in sociotechnical systems remains an open challenge. For instance, DT modeling is recognized as of paramount importance for its prosperous utilization, but there is not a general agreement in the community on a general approach to build the DT models [21]. The implementation of DT in domains such as the Cyber-physical fusion also remains an engineering challenge due to the lack of a generally accepted framework for acquiring the data, transmit the data, building operative knowledge from data through simulation or data mining, and controlling the CPS [14]. Only recently, researchers have started focusing on the architectural aspects of DT [16]. For example, in [2] authors propose a layered architecture reference model for cloud-based CPS, in [17] authors propose a four-layer architecture pattern to design DT incorporating various types of information sources. The pattern is designed to be extensible with respect to the number of sources, and flexible with respect

to the type of information. However, in spite of the increasing interest in DT, little effort has been devoted to investigate architectural quality attributes, such as modifiability, scalability, evolvability, reusability.

Architecture Alignment – The need and importance for aligning the Organizational and IT architectures as been pinpointed over time by many research studies [7][22], and now many points of alignment are commonly recognized. Early studies focused on linking the business plan with the IT plan, and/or on ensuring the alignment of business and IT strategies. Other approaches instead examined the mapping between business needs and information system priorities. The implications of architecture alignment have been also demonstrated through many empirical case studies (e.g., [10][15]). The results demonstrate that the organizations that have their Organizational and IT strategies aligned outperform those organizations that do not. Indeed, alignment leads to more focused and strategic use of IT which, in turn, leads to increased performance [6].

Architecture Evolution – Sociotechnical systems obey the principle of *there is nothing permanent except change*. The Organizational/Business Architecture will evolve and the Information architecture needs to keep aligned. In consequence, the DTO of the sociotechnical system needs to evolve too. At present, evolving the software-based DT is possible, but doing it in a systematic way that does not boost the software engineering expenses resides in the research frontier. In [18], the authors propose a pattern-oriented development approach, where patterns are considered as the main building blocks of the architecture and changes are applied by means of patterns substitution, i.e., design evolution is identified in terms of replacement of patterns by other patterns. Another interesting approach is proposed in [3], where the authors present the Evolution Style, which defines a family of domain-specific architecture evolution paths that share common properties and satisfy a common set of constraints. The evolution style specifies the set of concepts needed to define and analyze the software architecture evolution: (*i*) the set of operators defining the evolution transitions, (*ii*) the set of evolution path constraints defining whether a path is allowed or not, and (*iii*) the set of evaluation functions used to compare different evolution paths with respect to quality metrics.

5 Conclusions and Future Work

A *Digital Twin of the Organization* is an accurate representation of an organization (including physical, software, and human elements), which aims at facilitating decision-making processes, and making informed decision in the value chain. However, the lack of engineering best practices for developing and operating DTOs makes industry difficult to fully benefit from this technology. Further, as current solutions are often context-dependent or system-specific, they result challenging to change, extend and reuse.

In this paper we reported an initial attempt made in architecting and developing a DTO from a manufacturing industry, and its further evolution. In

particular, (*i*) we detailed a development process leveraging *Enterprise Architecture* to map the organization structure/behavior (i.e., Organizational/Business Architectures) and status (i.e., Information Architecture) into a DTO, (*ii*) we pinpointed a set of concerns to be confronted with when architecting a DTO, and (*iii*) we discussed the need for an Architectural Framework enabling the specification, alignment, and evolution of OA/BA, IA and DTO.

To this end, as a future work we mainly plan to address the aforementioned architectural concerns, by devising an architectural framework which specifically takes into account DTO's Modularity, Granularity, Decomposition/Composition, and Evolution.

References

1. Aitken, A.: Industry 4.0: Demystifying digital twins, https://www.lanner.com/
2. Alam, K.M., El Saddik, A.: C2ps: A digital twin architecture reference model for the cloud-based cyber-physical systems. IEEE Access **5**, 2050–2062 (2017)
3. Barnes, J.M., Garlan, D., Schmerl, B.: Evolution styles: Foundations and models for software architecture evolution. Softw. Syst. Model. **13**(2), 649–678 (May 2014)
4. Bass, L., Clements, P., Kazman, R.: Software Architecture in Practice. Addison-Wesley Professional, 3rd edn. (2012)
5. Baxter, G., Sommerville, I.: Socio-technical systems: From design methods to systems engineering. Interact. Comput. **23**(1), 4–17 (Jan 2011)
6. Chan, Y.E., Sabherwal, R., Thatcher, J.B.: Antecedents and outcomes of strategic is alignment: an empirical investigation. IEEE Transactions on Engineering Management **53**(1), 27–47 (Feb 2006)
7. Chan, Y.E., Reich, B.H.: It alignment: what have we learned? Journal of Information Technology **22**(4), 297–315 (Dec 2007)
8. Chen, P., Han, J.: Facilitating system-of-systems evolution with architecture support. In: Proceedings of the 4th International Workshop on Principles of Software Evolution. pp. 130–133. IWPSE '01, ACM, New York, NY, USA (2001)
9. Cognizant: Is your organization ready to embrace a digital twin?, https://www.cognizant.com/
10. de Leede, J., Looise, J., Alders, B., Alders, B.: Innovation, improvement and operations: an exploration of the management of alignment. International journal of technology management **23**(4), 353–368 (2002)
11. El Saddik, A.: Digital twins: The convergence of multimedia technologies. IEEE MultiMedia **25**(2), 87–92 (Apr 2018)
12. Grieves, M.: Digital twin: Manufacturing excellence through virtual factory replication. White paper pp. 1–7 (2014)
13. Herraiz, I., Rodriguez, D., Robles, G., Gonzalez-Barahona, J.M.: The evolution of the laws of software evolution: A discussion based on a systematic literature review. ACM Comput. Surv. **46**(2), 28:1–28:28 (Dec 2013)
14. Josifovska, K., Yigitbas, E., Engels, G.: Reference framework for digital twins within cyber-physical systems. In: Proceedings of the 5th International Workshop on Software Engineering for Smart Cyber-Physical Systems. p. 2531. SEsCPS 19, IEEE Press (2019)
15. Kearns, G.S., Lederer, A.L.: A resource-based view of strategic it alignment: How knowledge sharing creates competitive advantage. Decision Sciences **34**, 1–29 (2003)

16. Malakuti, S., Grüner, S.: Architectural aspects of digital twins in IIoT systems. In: Proceedings of the 12th European Conference on Software Architecture: Companion Proceedings. ECSA 18, Association for Computing Machinery (2018)
17. Malakuti, S., Schmitt, J., Platenius-Mohr, M., Grüner, S., Gitzel, R., Bihani, P.: A four-layer architecture pattern for constructing and managing digital twins. In: Proceedings of the 13th European Conference on Software Architecture. pp. 231–246 (2019)
18. Ram, D., Rajasree, M.: Enabling design evolution in software through pattern oriented approach. In: Konstantas, D., Lonard, M., Pigneur, Y., Patel, S. (eds.) Object-Oriented Information Systems, Lecture Notes in Computer Science, vol. 2817, pp. 179–190. Springer (2003)
19. Schluse, M., Priggemeyer, M., Atorf, L., Rossmann, J.: Experimentable digital twinsstreamlining simulation-based systems engineering for industry 4.0. IEEE Transactions on Industrial Informatics **14**(4), 1722–1731 (April 2018)
20. Sousa, P., Caetano, A., Vasconcelos, A., Pereira, C., Tribolet, J.: Enterprise architecture modeling with the unified modeling language. In: Rittgen, P. (ed.) Enterprise Modeling and Computing with UML, pp. 67–94. IGI Global (2007)
21. Tao, F., Zhang, H., Liu, A., Nee, A.Y.C.: Digital twin in industry: State-of-the-art. IEEE Transactions on Industrial Informatics **15**(4), 2405–2415 (April 2019)
22. Ullah, A., Lai, R.: A systematic review of business and information technology alignment. ACM Trans. Manage. Inf. Syst. **4**(1), 4:1–4:30 (Apr 2013)