Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 232 (2024) 1089–1098

Procedia Computer Science

www.elsevier.com/locate/procedia

5th International Conference on Industry 4.0 and Smart Manufacturing

Digital twin architecture for assembly line performance monitoring

Andrea Bonci[a,*], Alessandro Di Biase[a], M. Cristina Giannini[a], Sauro Longhi[a], Mariorosario Prist[a]

[a]*Dept. of Information Engineering (DII), Politechnic University of Marche, 60131 Ancona, Italy*

Abstract

The forthcoming industrial requirements to ensure increasingly flexible and agile production processes will require an ever-increasing ability to recognise, predict and optimise the state of processes with respect to the planned state. Advanced tools such as Digital Twins (DT) and Cyber Physical Systems (CPS) can help manage and address these problems. This paper presents some results of the authors' research on the adoption of a basic methodology for the design and implementation of Digital Twins aimed at manufacturing applications and subsequent integration into a Cyber Physical Production Systems (CPPS). Reference publications were selected based on the level of detail of DT and CPS architectures. The aim is also to combine methodological development with industrial tools and requirements. The architecture is potentially applicable to different production scenarios. Finally, the first results of application on a scale production line are shown.

Keywords: Digital Twin, Industry 4.0, Industrial Internet Reference Architecture, Cyber Physical System

1. Introduction

Nowadays, companies are always looking for solutions to increase productivity, improve quality and reduce costs. Implementing digital manufacturing solutions and increasing digitisation of products, making them capable of exchanging or processing information, is one of the key initiatives in this journey. The aim is to transform the conventional manufacturing approach into a digitally enabled manufacturing. To this end, DT and CPS are technologies that will enable the transition and fulfilment of customer requirements. There are many definitions of DT in the literature, and just as many descriptions of how it can be used. The same happens for the CPS. The origin of the concepts of DT and CPS can be traced back to Grieves [1] and Lee [2] respectively. In its original form the DT was described in the PLM (Product Lifecycle Management) context as "a virtual representation of what has been produced". Other definitions of DT have followed over time and a fairly comprehensive review can be found in [3]; these mainly refer to the description of the DT as a virtual representation capable of functioning on different simulation disciplines synchro-

* Corresponding author. Tel.: +39-071 220 4666.
E-mail address: a.bonci@univpm.it

1877-0509
10.1016/j.procs.2024.01.107

nised with real system thanks to sensed data and connected smart devices, mathematical models and data processing. Furthermore, in accordance with a more recent and broader definition [4] the DT is "a virtual representation of a physical object or system across its life cycle, using real-time data to enable understanding, learning and reasoning". On the other side, following the first coining of the word CPS by H. Gill to describe increasingly complex systems that could not be effectively illustrated using traditional IT terminology [5], and in accordance to the original definition of E. A. Lee [2, 6], CPSs "are integration of computation and physical processes. Embedded computers and networks monitor and control the physical processes, usually with feedback loops where physical processes affect computations and vice versa". In other words, CPSs aim to "monitor and control the physical processes with feedback loops". Hence, we can infer that the "feedback control" capabilities of CPSs is beyond the scope of DTs, while with them they may share the monitoring, the cyber-physical integration and some kind of reasoning aspects. Since their conception, CPSs have been proposed for a wide range of applications in manufacturing, ranging from production planning and control systems [7, 8, 9] to logistic systems [10, 11] and many others [12, 13]. Similarly, DTs have been proposed in many forms and applications in the manufacturing sector [3], ranging from production planning and decision support systems [14] to DTs for the shop-floor [15], or parts thereof [16, 17, 18], or even DTs for maintenance using Machine Learning techniques [19, 20, 21] rather than signals [22, 23] or others techniques. More recently DTs have also been proposed for the automotive industry, in particular to physical two-wheelers [24] whose complex dynamics, investigated in [25, 26], clearly require quite different computational performance and requirements. As we have seen, the development of DTs is highly dependent on the field of application, so it is useful to study both a methodological approach possibly based on reference architectures and practical industrial tools for DT development. This paper has a threefold objective:

- First, it proposes a methodological approach based on Industrial Internet Reference Architecture (IIRA) [27] for the design and implementation issues of a DT for monitoring and analysing the performance of manufacturing lines in order to implement subsequent planning and control to increase productivity.
- Second, it identifies a set of industrial architectural tools, mainly focused on the interoperability, extendibility, scalability, deployment, etc. which are useful for developing the implementations. This because a DT for industrial applications, which also wants to be open to innovation, requires an architecture that meets both industrial guidelines and the methodological requirements.
- Finally, a preliminary application example is presented that introduces a method for developing a DT of an assembly line based on finite state machines in Python software.

The rest of this paper is organized as follows. An introduction of DT development approaches from both methodological and industrial perspective are described in Section 2. An hybrid architecture for combining different approaches is introduced in Section 3. The preliminary proof of concept application of the proposed approach is presented in Section 4. Remarks and future works conclude the paper.

2. Digital Twin development approaches

This section presents the design approaches available in the literature used to develop DT, both from a methodological point of view and the use of standard industrial architectures. A complete review of architectures is not the goal of this paper, so only a few of those considered by the authors to be the most interesting for the purpose will be described. Both of the above approaches will later be merged into a hybrid development methodology.

2.1. Digital Twin - Methodological Approaches

Several approaches for the development of DTs have been proposed in literature to support the DT design from methodological point of view; in this section, only a few are listed, such as the approach of Park et al. [28] which presents a formal method based on a four-layer DT architecture consisting of the layers 1.design, 2.network, 3.development, and 4.verification. A further approach from Qamsane et al. [29] proposes a baseline framework for DT life-cycle that illustrates how to divide the DT life-cycle process into two main phases which are off-line development and on-line deployment and maintenance. Specifically, five off-line phases were identified: 1.envision (identify

the high-value problem, requirements, objective, etc.), 2.design (define a solid architecture, data collection, modeling approach, etc.), 3.develop (DT model development, model training, etc.), 4.verify (the solution must address the problem), and 5.validate (validate the solution on the problem). Six online phases are also identified: 1.deploy (integration, control rules, test, evaluate, etc.), 2.use (the DT meets the expected results), 3.evaluate (periodic evaluation of DT benefits), 4.maintain (check that the DT provides the expected benefits), 5.tune (online update of DT mainetenance), and 6.rebuild (determine if current design is adequate). Ariansyah et al. [30] proposed more complex guideline covering the initial phase of the DT and the modelling of all elements through the validation phase in order to verify whether the purpose of the DT has been achieved. Their eight layers method was organized in 1.define the requirements, 2.develop the information flow, 3.develop the virtual representation, 4.develop the data exchange system, 5.develop the analysis method, 6.develop actions, 7.verify and 8.validation. On the other hand, the methodology proposed by Heindl et al. [31], in a similar way of Qamsane et al., attempts to cover the entire life-cycle of the DT design phase, from idea to deployment. They propose an eleven levels solution composed by: 1.envision, 2.design, 3.develop, 4.verify, 5.validate, 6.deploy, 7.use, 8.evaluate, 9.maintain, 10.tune, 11.rebuild. As far as the DT development methodology in the manufacturing sector is concerned, no specific guidelines for manufacturing sector can be derived from the above analysis. This leads to a lack of guidelines from the point of view of the implementation phase for manufacturing sector. There seems to be a gap between the methods proposed in literature from a methodological point of view and the methods that can be useful for manufacturing applications which require standard industrial approaches and industrial-ready reliable tools, For this reason, in order to have a complete overview of all the steps needed to implement a DT in the manufacturing sector, thus capable of being interconnected with the IT infrastructure of a factory, this paper proposes the use of the Industrial Internet reference architecture (IIRA) [27], which is specialised in implementing and applying the enabling technologies of Industry 4.0 to industries. It also includes the design of a DT for industry. The next subsection will present a brief introduction of the IIRA, with a focus on the industrial standard architectures that comprehensively define DT development.

2.2. Digital Twin - Industrial standard architectures

The industrial approach to the design and development of a DT can be based on the Industrial Internet reference architectures [27] which is a way to standardise both how IoT systems are implemented in different domains and, among other things, how a DT is designed. Breivold [32], for example, has studied many different reference architectures, paying attention to those that enable the integration of the IoT into an IT infrastructure, including the DT, e.g. the standard IEEE P2413 [33], the Industrial Internet Reference Architecture (IIRA) [27], the reference architectural model for Industry 4.0 RAMI 4.0 [34], the reference architecture for the Internet of Things WSO2 IRA [35], etc. Among the various architectures presented by Breivold, only a few have an explicit and sufficiently detailed definition of how to build a DT, e.g. the IIRA, which presents an architectural framework for developing inter-operable IIoT systems for different applications at different industrial levels. It lays down guidelines for building a DT or a set of DTs in an hierarchical structure. In this context, IIRA proposes twelve detailed steps: 1.characteristics of a DT, 2.relations among DTs, 3.role of DTs, 4.DT design , 5.information modeling, 6.information population, 7.information synchronization, 8.APIs, 9.connectivity, 10.deployment, 11.security, 12.interoperability. Both the methodological approaches and industrial architectures, respectively in Sections 2.1 and 2.2 deal with the DT topic but focus on different aspects; therefore, a hybrid architecture is proposed in the next section to act as a glue between these aspects.

3. Hybrid development methodology for Digital Twin

Combining the needs of the methodological approaches and the use of industrial architectures shown above is not trivial as these deal with the same topic (DT) but focus on different aspects without taking others into account.

Indeed, methodological approaches available in the literature focus mainly on the mathematical and formal modelling of the DT, and much less or not at all on how it can be implemented on the Information Technology (IT) infrastructure of a company or, again, on how it can be interconnected with the other systems that are key players in the decision-making chain. On the other hand, the industry-standard architectures only define how to build a modular, flexible, stable, extensible and shareable DT system, and are deficient in providing ways to support the modelling phase of the DT model. In this context, from the industry's point of view, the IIRA standard was chosen as a reference

due to the fact that it not only provides for the development phases of the DT, but also provides the architectural shell for the exchange of data between many other actors in the factory ecosystem and also with other DTs. While from a methodological point of view, Heindl's methodological model might be preferred for its completeness, its suitability for an industrial implementation is not so obvious. For this reason, by taking the IIRA standard as a reference and combining it with the main aspects of Heindl's model, the authors propose a useful compromise for developing and implementing a DT in the manufacturing environment. This approach proposes a DT implementation process consisting of six macro-phases. Each macro-phase is associated with one or more blocks characteristic of both Heindl's model and the chosen industrial architecture (IIRA). The macro-phases identified for the development of the DT and resulting from the fusion of the characteristic aspects of both the Heindl model and the IIRA architecture are summarised in Figure 1 and are described and detailed individually below. The fusion of both architectures is based on the similarity analysis and comparison of each stage as proposed by the respective authors.

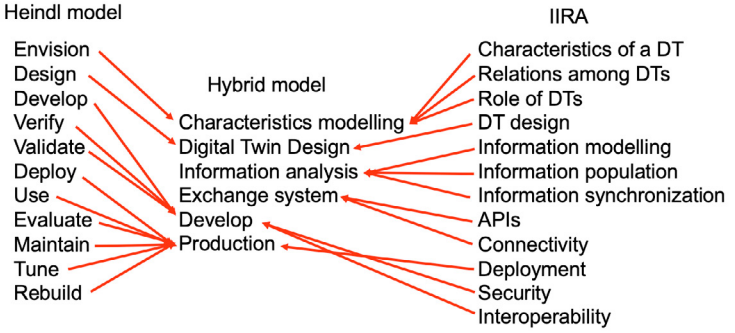

Fig. 1: Hybrid development methodology for Digital Twin.

1. *Characteristics modelling*
   The procedure: It combines the first level of Heindl model and the first three levels of the IIRA architecture: (i) define the characteristics of a DT, (ii) the relations among DTs, (iii) role of DTs in the life cycle of entities and, (iv) define the requirements and Envision). See Figure 1.
   The tasks: are performed through the collection of requirements using different techniques and tools.
   The tools: use of Flowcharts software to depict sequential flow and control logic of a related set of activities, Gantt Charts to provide a visual representation of tasks along with their scheduled timelines, Gap analysis tool to evaluates the gaps in a product's performance to determine whether requirements are met or not, etc.
2. *Digital Twin Design*
   The procedure: It combines the second level of Heindl model and the fourth level of the IIRA architecture: (i) DT internal design and, (ii) build the virtual representation and Design. See Figure 1.
   The tasks: the creation of a DT is synonymous with the creation of a "system level model", i.e. a model that incorporates the dynamics of several subsystems together, often including multiple engineering domains.
   The tools: Virtual and computational models can be modelled using physical models, statistical models, control or transition models, machine learning models, rule and behaviour models, etc. [36, 37]. In this context, there are many tools available on the market such as IBM Rhapsody, Siemens NX, Autodesk, Ciros, Microsoft Azure solution based on DTDL or a custom software developed using Python, C++, etc.
3. *Information analysis*
   The procedure: It combines only from the fifth to the seventh level of the IIRA architecture because the Heindl model does not take into consideration how information have to be modeled and how data is exchanged in and out of the DT: (i) information modeling, (ii) information population and, (iii) information synchronization. See Figure 1).
   The tasks: are about the design of mechanisms to structure and make the content of DT modular, and to extend the content when new types of information become available, furthermore, the design of how to populate the DT with information and finally how to synchronise information between one DT and another.
   The tools: the focus is on how to design the databases, for saving in real-time both the data streaming coming

from the field and the results of the analysis. This phase starts with the requirements analysis in the first step, through the entity-relationship diagram to the interfaces to exchange the data. Some available tools are Microsoft SQL Server, MySQL Workbench, Amazon DynamoDB, IBM microservice, Azure Data Explorer cluster, etc.

4. *Exchange system*
   The procedure: It combines the eighth and ninth level of the IIRA architecture, this because the Heindl model does not take into consideration how to exchange data in or out a DT. Therefore it can be summarized in: (i) APIs and, (ii) connectivity. See Figure 1.
   The tasks: the APIs are an important aspect for reading and writing data, while connectivity is the key factor for interactions between the DT and the real plant and for interaction with and among DT. *The tools:* could be realised by implementation of RESTfull architecture, MQTT, WebSocket, OPC-UA, etc.
5. *Develop*
   The procedure: It combines the third, the fourth and the fifth level of Heindl model and the eleventh and twelfth of the IIRA architecture. See Figure 1. Despite Heindl model, IIRA does not take into consideration an explicit level of how to develop a DT because it is intrinsic in the its fourth level. Typically it also covers (i) the architectural design which is how and where the DT will be held for the production stage, (ii) the security due to the fact that the DT could interact with many actors that have different mechanisms to secure access and (iii) the validation of the all requirements.
   The tasks: manages the software development life-cycle which is step-by-step process bringing a DT concept from the envision stage to implementation and eventually to the market.
   The tools: public or private cloud with firewall and authentication manager (Amazon, IBM, Google, Microsoft Azure DTDL, etc.).
6. *Production*
   The procedure: It combines the tenth level of the IIRA architecture and from sixth to eleventh level of Heindl model. See Figure 1.
   The tasks: loading the developed software from the staging area to the production area. This is the last step for enabling the people to use the whole system during the working activities.
   The tools: Any type of public or private cloud infrastructure with the possibility to manage the staging and production area (Amazon, IBM, Google, Microsoft Azure DTDL, etc.).

The hybrid DT development methodology described above facilitates the development of the DT in production while also complying with industry standards. In order to evaluate the proposed methodology, a preliminary proof of concept for the development of a DT for an assembly line is summarised in the following section

4. Preliminary proof of concept study

Preliminary experiments were carried out to verify the application of the proposed hybrid methodology from the requirements of a real use case consisting of an automated assembly line, in order to develop all six macro-phases of the hybrid architecture. This paper examines the first two in detail as application examples, leaving the later ones for future development, while providing some general insights for the latter. Referring to the six steps of the above section we will have:

1. *Characteristics modelling:*
   Objective: To verify the effectiveness of the hybrid development methodology for DT, a contextualized proof of concept was conducted on a real assembly line. The *Characteristics modelling*, focused on the collection of requirements, is the first step of the DT development in terms of service and business organizations. The idea, for the proof of concept, is to design the DT of an industrial asset based on the dynamic evolution of performance indicators (KPIs) built from data. These can be used to monitor performance from both machines and labours [38], detect early deterioration in performance and enable preventive maintenance, as well as to identify constraints and perform “what-if” analyses to make informed decisions.
   Tools: a production line located in Italy in a Public-Private-Partnership laboratory named i-Labs Industry was used for experiments. The assembly line was manufactured by Festo and it is shown in Figures 2 and 3.

Fig. 2: Assembly line by Festo.

It performs the assembly process of the main components of simplified mobile phones, made of three components, the front cover, the back cover and a Printed Circuit Board (PCB). It consists of four physical stations with six different modules: (i) AR/SR an automatic warehouse, (ii) Pick-by-Light module for manual work, (iii) Camera 1 module for quality control, (iv) manual rework station, (v) Camera 2 module for final quality control, and (vi) cover loader module (See Figure 3 to know the correspondence between the numbers and the workstations).

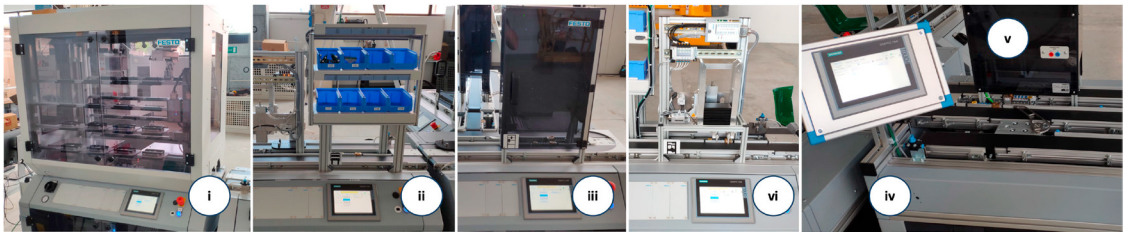

Fig. 3: Assembly line modules.

The six modules are managed by PLCs, interconnected by hardware and software coupling [39]. The interconnection with the external world is based on the OPC-UA protocol. Each module is equipped with an OPC-UA server that communicates with a MES provided by Festo and named MES4; integration with the DT takes place at a higher level using the data available in the MES4 database.
Considering the goal of modeling the manufacturing process to analyze its performance, and having only state and sensor variables as data available from the assembly line modules, it is relevant to the development of the DT to have an abstraction layer that describes the system with states and transitions or, in other words, by means of a finite state machine (FSM) to track the evolution of the system in the digital world. With this perspective it is useful to have a list of the production states and all the transitions that allow the FSM to evolve from one state to another. A Finite State Machine is a computational abstraction where the program behaviour depends on its current state and the input it receives. The machine can perform a transition to a different state based on inputs or events. The evolution and design of the FSM is based on the data log of the MES4 which includes states and transitions such as: *tblMaintError* (list of errors), *tblFinStep* (steps performed by each module), *tblMachineReport* (list of state changes), *Tu* (equipment running time), etc. Delving deeper into the production process, three main production situations of the whole assembly line can be identified:

- First situation - *No defects*, the assembly line is producing only good items:
  (i) AR/SR, (ii) Pick-by-Light, (iii) Camera 1 - ok, (iv) Cover Coader, (v) Rework - Cover, (vi) Camera 2 and (vii) AR/SR.
- Second situation - *Repairable defects*, assembly line produces repairable items through manual workstation:
  (i) AR/SR, (ii) Pick-by-Light, (iii) Camera 1 - ko, (iv) Rework - fix, (v) Camera 2, (vi) Cover Loader, (vii) Rework - cover, (viii) Camera 2, and (ix) AR/SR.

- Third situation - *Irreparable defects*, assembly line produces irreparable item which is discarded at the end of the process:
  (i) AR/SR, (ii) Pick-by-Light, (iii) Camera 1 - ko, (iv) Rework - fix, (v) Camera 2, and (vi) AR/SR.

These situations can clearly be sourced from the MES system.

WPNo	StepNo	ONo	OPos	OpNo	NextStepNo	FirstStep	ErrorStepNo	NewPNo	PlanedStart	PlanedEnd	Start	End
1210	50	2178	2	510	60	FALSO	0	0	23/06/23 10:55	23/06/23 10:55	23/06/23 10:57	23/06/23 10:57
1210	50	2178	1	510	60	FALSO	0	0	23/06/23 10:55	23/06/23 10:55	23/06/23 10:57	23/06/23 10:57
1210	40	2178	8	201	50	FALSO	0	0	23/06/23 10:55	23/06/23 10:55	23/06/23 11:00	23/06/23 11:00
1210	40	2178	7	201	50	FALSO	0	0	23/06/23 10:55	23/06/23 10:55	23/06/23 11:00	23/06/23 11:00

Fig. 4: Data available from the MES database.

2. *Digital Twin Design:*
   Objective: The second goal of hybrid DT development is related to the realization of a "system-level model". The question here is how to represent the real plant digitally. The DT is positioned at the intermediate level between the field and the decision-making level. The most appropriate way in our opinion, as introduced in the previous subsection, is to use a Finite State Machine, implementing the relational rule model that establishes the interaction between product attributes and process sibling services, control codes and simulation flow.
   Tools: among various software libraries for designing a DTs and also useful for implementing FSM, like JavaScript State Machine, lightweight stateless4j Java, Python-statemachine, Ruby micromachine, the Python Transitions library has been here selected and used. This choice is due to the fact that it provides not only the ability to define (i) states and transitions for a linear or hierarchical structure, but also (ii) to attach callbacks to transitions as well as states in order to call methods before or after the transition execution, to use the asynchronous programming paradigm and, most importantly, to generate basic state diagrams displaying all valid transitions between states using the capabilities of the Graphviz library.
   Starting from the available log-file data provided by the MES database, here showed in Figure 4, the states and the transitions were listed. The identified states refer to the operating conditions of the assembly line and are directly related to the module that is currently active. Eleven states have been identified to model the system behaviour: state 0 (idle), state 10 (AR/SR module - Raw material), state 20 (Pick-by-Light), state 30 (Check camera 1), state 40 (Cover Loader), state 50 (Rework - press parts together), state 60 (AR/SR module - good item), state 96 (Check camera 2), state 95 (Rework - Fix), state 97 (AR/SR module - Discarded item) and state 200 (Fault). The transition between two states can be handled either by an event, i.e. the result of a camera module, or by the state map described above, i.e. from the state 20 to the state 30 there is no event but the transition is directly enabled (there is only a delay due to the movement of the item using the conveyor belt).
   In Table 1 are highlighted the time to transport the carrier from one module to another and the working time spent in a module, while the complete FSM in Python that implements the behaviour of the assembly line is highlighted in Figure 5 generated by Graphviz library.

Table 1: Transport time and Working time between brackets.

	AS/RS	Pick-by-Light	Camera 1	Cover Loader	Rework - fix	Camera 2	Rework - press
AS/RS	(10.6 s)	13.1 s					
Pick-by-Light		(25.2 s)	12.5 s				
Camera 1			(1.1 s)	26.3 s	26.8 s		
Cover Loader				(4.5 s)			78 s
Rework - fix					(36.3 s s)	12.2 s	
Camera 2	26.5 s			15.9 s		(1.2 s)	
Rework - press						21.4 s	(14.7 s)

Below we will summarize the actions deemed necessary to also develop the additional four points of the hybrid DT development for the assembly line proof of concept. These will be the subject of future works, for which we

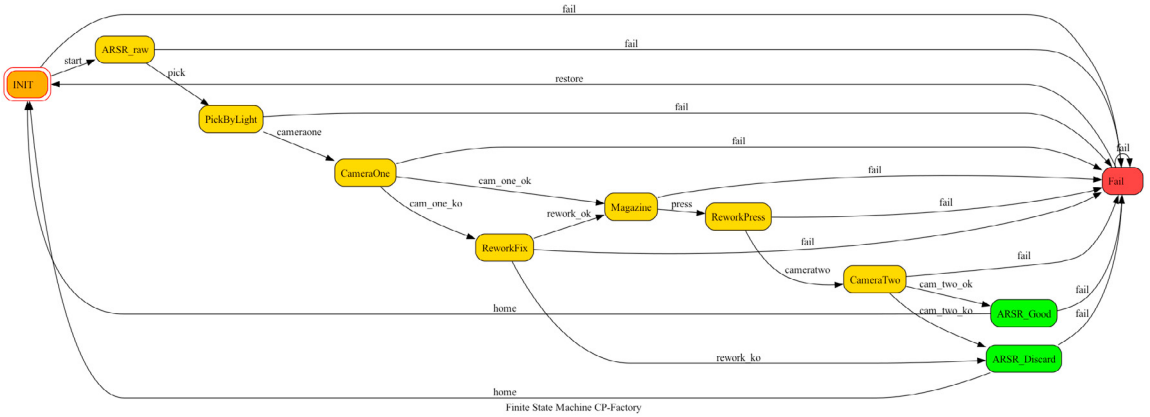

Fig. 5: FMS Graphviz diagram.

still provide some guidelines for development.

3. *Information analysis:*
   Objective: This step defines the structure and content of DT to facilitate the exchange of information. Data population is achieved via APIs.
   Tool: offline populating of data can be achieved by using a predefined serialisation format such as JSON or by using an information synchronisation mechanism between DTs. In our case a set of APIs directly correlated with the states and transitions of FSM has been designed using a JSON formalism (See Code 1). In the JSON formalism have been defined different type of classes (one for each state) having parameters such as the *PickByLight* class with parameters *Definition* and *Properties*. In detail, the APIs are able to *Run and stop the FSM*, *Create and query the states*, *Get and update states*, *Publish data messages* and *Get and Set parameters*.

Listing 1: Extract of JSON file.

```
"PickByLight": {
  "definition": [
    "state:SwitchableState:1.0.0",
    "state:Switchable:1.0.0"
  ],
  "properties": {
    "configuration": {
      "on": false,
      "changeDate": "2023-06-01T16:42:23Z",
      "reportEvents": true,
      "currentEventsInterval": 30
    },
    "status": {
      "currentState": 0
    }
```

4. *Exchange system:*
   Objective: this layer is direct correlated with the third step in term of architecture to integrate different sources of data and the way to share information. As introduced in the previous section, this layer is necessary to manage heterogeneous data sources and control the interaction with other information sources to enrich and to synchronise the DT content.
   Tools: APIs are used to interact at different levels, such as devices, edge or cloud levels, and some mechanisms to uniquely identify the DT must be provided, Figure 6 summarises the interconnections provided for the exchange system of the assembly line considered. In fact the DT, running on a dedicated company PC, communicates from one side with the *Service* (used to read data form the MES4 and to fire events in the DT), and from the other side with the *Cloud*, the external supervisor system in order to apply analysis algorithms.
5. *Develop:*
   Objective: it is devoted to program the entire system, starting from the FSM, which consists of the model and the simulation interface (in our case Python environment), and by the other tools, i.e. the input and output signals of the assembly line modules, as well as the check functions and functional groups.

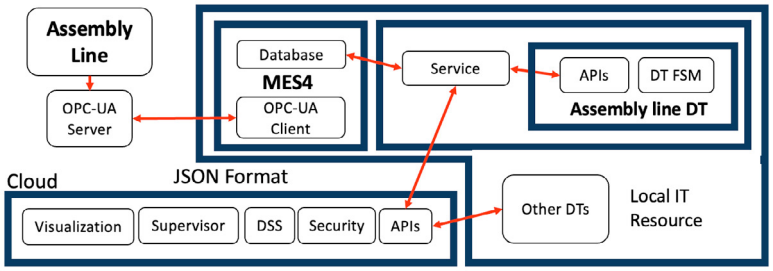

Fig. 6: Exchange system interconnections.

Tool: Python programming can be used both to manage the behaviour of the simulation interface and to implement the RESTfull architecture.

6. *Production:*
   Objective: in this last step, the deployment of the DT on the corporate IT infrastructure must be analysed. It consists in the testing of the software on a real system and in its further testing by the users, this phase will also include integration with other programs such as decision support systems (DSS). At this stage, it is important to assess latency requirements and response times, as well as integrability with other systems.
   Tool: the use of Windows or Linux servers tools could be a possible solution to run the DT interface program but for the proof of concept a DigitalOcean's Cloud has been chosen. It is a cloud platform that supports IoT or edge business, with a focus on building and scaling technology.

5. Conclusion and future works

Various research in the field of DT shows the effectiveness of integrating DT into the product-process, highlighting the commercial benefits and encouraging its adoption in existing industrial infrastructures. Although there are different approaches to DT design for industrial applications, the methodological approaches proposed in the literature to develope DTs and the industry standards useful for DTs implementation have different purposes albeit applied to the same concept of DT. To bridge this gap, this paper proposes a hybrid architecture developed in six macro-phases and presents a preliminary proof of concept study in which two of the six phases were developed. In this context, a DT of an assembly line based on finite state machines in Python software has been presented. Future work will focus on the development of the remaining phases.

Acknowledgements

This research was funded by Horizon Europe project "EDIH4Marche" (European Digital Innovation Hub for Marche), Call DIGITAL-2021-EDIH-01, G.A. no. 101084027 topic DIGITAL-2021-EDIH-INITIAL-01.

References

[1] M. Grieves, "Digital Twin: Manufacturing Excellence through Virtual Factory Replication," Whitepaper, 2014.
[2] Edward A. Lee (2007) "Computing Foundations and Practice for CyberPhysical Systems: A Preliminary Report," *Technical Report No. UCB/EECS-2007-72 - Electrical Engineering and Computer Sciences University of California at Berkeley* pp 1-25
[3] Werner Kritzinger, Matthias Karner, Georg Traar, Jan Henjes, Wilfried Sihn, (2018) "Digital Twin in manufacturing: A categorical literature review and classification", *IFAC-PapersOnLine*, Volume 51, Issue 11, Pages 1016-1022,
[4] "NIC UK: Nat. Infrastr. Commiss. UK". [Online]. https://nic.org.uk/app/uploads/Data-for-the-Public-Good-NIC-Report.pdf, access: July 2023.
[5] Gill H. (2006) "NSF perspective and status on cyber–physical systems" *NSF Workshop on Cyber–physical Systems*, Oct 16–17; Austin, TX, USA. Alexandria: National Science Foundation.
[6] Edward A. Lee (2008) "Cyber Physical Systems: Design Challenges." *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC).* (): 363-369.
[7] J. Shi, J. Wan, H. Yan and H. Suo, "A survey of Cyber-Physical Systems," 2011 International Conference on Wireless Communications and Signal Processing (WCSP), Nanjing, China, 2011, pp. 1-6,

[8] A. Bonci, M. Pirani, and S. Longhi. (2017) "An Embedded Database Technology Perspective in Cyber-physical Production System." *Procedia Manufacturing.* Vol. 11: pp 830–837.

[9] A. Bonci, M. Pirani and S. Longhi, "A Database-Centric Framework for the Modeling, Simulation, and Control of Cyber-Physical Systems in the Factory of the Future," (2018), *Journal of Intelligent Systems*, vol. 27, num. 4, pp 659–679,

[10] L. Cavanini, P. Cicconi, A. Freddi, M. Germani, S. Longhi, A. Monteriu, E. Pallotta, and M. Prist. (2018) "A Preliminary Study of a Cyber Physical System for Industry 4.0: Modelling and Co-Simulation of an AGV for Smart Factories." *2018 Workshop on Metrology for Industry 4.0 and IoT.* pp 169-174.

[11] P. Zheng, H. Wang, Z. Sang, RY Zhong, Y. Liu, C, Liu et al. (2018) "Smart manufacturing systems for Industry 4.0: conceptual framework, scenarios, and future perspectives." *Front Mech Eng.* 13(2):137–50.

[12] Liu, Yang and Peng, Yu and Wang, Bailing and Yao, Sirui and Liu, Zihe. (2017) "Review on cyber-physical systems." *IEEE/CAA Journal of Automatica Sinica.* (4): 27–40.

[13] Rainer Stark, Carina Fresemann and Kai Lindow. (2019) "Development and operation of Digital Twins for technical systems and services." *CIRP Annals.* (10): 129-132.

[14] R. Rosen, G. von Wichert, G. LoK. D. Bettenhausen (2015). "About the Importance of Autonomy and Digital Twins for the Future of Manufacturing."*IFAC-PapersOnLine,* 48 (3), pp. 567– 572.

[15] T. Fei and Z. Meng. (2017) "Digital Twin Shop-Floor: a new Shop-Floor Paradigm Towards Smart Manufacturing." *IEEE Access.* 20418-20427.

[16] Yun, Seongjin and Park, Jun-Hong and Kim, Won-Tae. (2017) "Data-centric middleware based digital twin platform for dependable cyber-physical systems." *Ninth International Conference on Ubiquitous and Future Networks (ICUFN).* (3): 922-927.

[17] T. H.-J. Uhlemann, C. Schock, C. Lehmann, S. Freiberger, R. Steinhilper, (2017). "The Digital Twin. Demonstrating the Potential of Real Time Data Acquisition in Production Systems." *Procedia Manufacturing,* 9, pp. 113–120.

[18] Konstantindis, Fotios and Gasteratos, Antonios and Mouroutsos, Spyridon G. (2018) "Vision-Based Product Tracking Method for Cyber-Physical Production Systems in Industry 4.0." *IEEE International Conference on Imaging Systems and Techniques (IST).*, (2): 1-6.

[19] R. Van Dinter, B. Tekinerdogan and C. Catal. (2022) "Predictive maintenance using digital twins: A systematic literature review."

[20] D. M. D'Addona, A. M. M. S. Ullah, D. Matarazzo, (2017). "Tool-wear prediction and pattern-recognition using artificial neural network and DNA-based computing." *J Intell Manuf,* 28 (6), pp. 1285–1301.

[21] G. A. Susto, A. Schirru, S. Pampuri, S. McLoone, A. Beghi (2015). "Machine learning for predictive maintenance: A multiple classifier approach." *IEEE Transactions on Industrial Informatics*, 11 (3), pp. 812–820. ' *Information and Software Technology.* (151): 107008.

[22] Bonci, Andrea and Longhi, Sauro and Nabissi, Giacomo and Verdini, Federica. (2019) "Predictive Maintenance System using motor current signal analysis for Industrial Robot." *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA.* (): 1453–1456.

[23] A. Bonci, S. Longhi,, G. Nabissi, (2021) "Fault Diagnosis in a belt-drive system under non-stationary conditions. An industrial case study,*Proceedings - 2021 IEEE Workshop on Electrical Machines Design, Control and Diagnosis, WEMDCD 2021*, pp. 260– 265,

[24] Jasper Smeets and Kemal Öztürk and Robert Liebich. (2023) "Digital twins for automotive development: Two wheelers application." *Advanced Engineering Informatics.* (56): 10-19.

[25] Bonci A., De Amicis R., Longhi S., Scala G. A. and Andreucci A. (2016) "Motorcycle lateral and longitudinal dynamic modeling in presence of tyre slip and rear traction." *21st Int. Conference on Methods and Models in Automation and Robotics (MMAR).* (): 391-396.

[26] Bonci A., De Amicis R., Longhi S., Lorenzoni E. and Scala G. A. (2016) "Motorcycle's lateral stability issues: Comparison of methods for dynamic modelling of roll angle." *20th Int. Conf. on System Theory, Control and Computing, ICSTCC 2016.* (): 607–612.

[27] "Industrial Internet Reference Architecture". [Online]. https://hub.iiconsortium.org/iira, last access: June 2023.

[28] Kyu Tae Park and Young Wook Nam and Hyeon Seung Lee and Sung Ju Im and Sang Do Noh and Ji Yeon Son and Hyun Kim. (2019) "Design and implementation of a digital twin application for a connected micro smart factory." *Int. J. of Computer Integrated Manuf.* (3): 596-614.

[29] Moyne, James and Qamsane, Yassine and Balta, Efe C. and Kovalenko, Ilya and Faris, John and Barton, Kira and Tilbury, Dawn M.. (2020) "A Requirements Driven Digital Twin Framework: Specification and Opportunities." *IEEE Access.*, (8): 107781-107801.

[30] Ariansyah, Dedy and Fernàndez del Amo, Iñigo and Erkoyuncu, John Ahmet and Agha, Merwan and Bulka, Dominik and De La Puante, Jose and Langlois, Marion and M'khinini, Yousra and Sibson, Jim and Penver, Steve. (2020) "Digital Twin Development: A Step by Step Guideline." *9th International Conference on Through-life Engineering Services (TESConf2020).* (2): 59-61.

[31] Heindl W. and Stary C. (2022) "Structured Development of Digital Twins A Cross-Domain Analysis towards a Unified Approach." *MDPI AG.* (10): 1490.

[32] Hongyu Pei Breivold. (2017) "A Survey and Analysis of Reference Architectures for the Internet-of-things." *ICSEA 2017 – The Twelfth International Conference on Software Engineering Advances.*.

[33] "IEEE P2413. Standard for an Architectural Framework for the Internet of Things (IoT). IEEE Standards Association". [Online]. http://https://sagroups.ieee.org/2413/, last access: June 2023.

[34] Roland Heidel. (2019) "RAMI 4.0." *DINDeutsches Institut für Normung e. V...*

[35] "A Reference Architecture For The Internet of Things". [Online]. https://wso2.com/whitepapers/a-reference-architecture-for-the-internet-of-things, last access: June 2023.

[36] Fei Tao, Bin Xiao, Qinglin Qi, Jiangfeng Cheng and Ping Ji. (2022) "Digital twin modeling." *Journal of Manufacturing Systems.*, (64): 372-389.

[37] Pires, Flávia and Melo, Victória and Almeida, João and Leitão, Paulo. (2020) "Digital Twin Experiments Focusing Virtualisation, Connectivity and Real-time Monitoring." *IEEE Conference on Industrial Cyberphysical Systems (ICPS).*, (2): 309-314.

[38] A. Bonci, D. Stadnicka, and S. Longhi, "The Overall Labour Effectiveness to Improve Competitiveness and Productivity in Human-Centered Manufacturing." In: *Trojanowska, J., Kujawińska, A., Machado, J., Pavlenko, I. (eds) Advances in Manufacturing III. MANUFACTURING 2022. Lecture Notes in Mechanical Engineering,* pp 144-155, Springer 2022.

[39] Bonci A., Di Biase A., Giannini M.C., Indri M., Monteriù A. and Prist M. (2022) "An OSGi-based production process monitoring system for SMEs." *IECON 2022 – 48th Annual Conference of the IEEE Industrial Electronics Society.*, (2): 1-6.