Digital Twins: Properties, Software Frameworks, and Application Scenarios

Roberto Minerva and Noël Crespi
Telecom SudParis, Institut Polytechnique de Paris

***Abstract*—The Digital Twin (DT) is an emerging approach that promises to change the way products and systems are made and used. The DT is attracting increasing interest in the Internet of Things community for its potential applications. Despite the hype, this approach requires a more precise definition and characterization in terms of its properties in relationship to software architectures and their platform implementations, as well as a deeper analysis of its potential applications and actual feasibility in several industries. This paper investigates the basic properties that hold for a DT, sketches a software framework and presents two application scenarios. The paper also addresses the business impact of DT by discussing servitization capabilities.**

The digital twin (DT) was conceived to design, prototype, and operate a virtual copy of a product.[1] It promotes the softwarization of physical objects (POs), into logical components and the development of applications that exploit programmability of the software representation. Originally used by NASA,[2] it has gained interest in manufacturing.[3] Because of its ability to describe POs by means of logical models and actual data, it is increasingly used in the Internet of Things and cyber-physical systems.[4] To fully master its potential, there is a need to understand what a DT is from a software perspective, how to support and implement it, how to manage the physical and logical counterparts, and what constraints are implied by the relationship between them. This article addresses the current DT definition, and identifies a set of characterizing properties. Next, a software framework is sketched out to illustrate two application scenarios that highlight technical possibilities and challenges. An analysis of some business benefits is provided. Conclusions summarize challenges and risks of this approach.

Digital Object Identifier 10.1109/MITP.2020.2982896

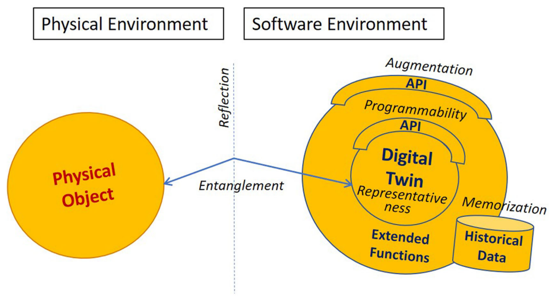

Figure 1. DT representation.

DIGITAL TWIN (DT) AND ITS PROPERTIES

DTs embody software capabilities to describe or simulate relevant features and behavior of POs from design to end-of-life phases.[5] A DT is a software actionable representation of a PO (or aggregated objects) in a specific environment. Some PO's features could be unrelated to the goals of the software representation. Products or complex objects are the main targets, because mapping one to one atomic objects may be unpractical or useless. DTs in a cyber-physical systems (CPS) domain are characterized[6] by events they are capable of managing, methods and functionalities they offer, storage capabilities to maintain a historical log of status changes, ability to serve as functional interfaces to applications through application programming interfaces (APIs) and ability to send/receive information. The definition of basic DT properties (see Figure 1) is instrumental to determine requirements and constraints posed to software platforms. The key properties of a DT are given as follows.

- *Representativeness and contextualization*. A DT represents the PO in a specific context, i.e., a DT is a software implementation of a model representing the PO in a specific environment. Feature simplifications or generalizations are possible when they do not affect the behavior in the specific context.
- *Reflection*. A DT mirrors the behavior and the status of the PO. Each change in status, each event faced by the PO is reflected by the DT. Changes that occur to the DT should be reproduced in the PO.
- *Entanglement*. A DT and its PO must be constantly "connected" to instantaneously (i.e., in a time period consistent with requirements of applications) register any change in status. Entanglement requires an effective, reliable communication between the PO and the DT suitable to the rate of changes.
- *Persistency*. A DT is always available. Its availability exceeds the actual existence of the PO. A DT could be available before the "creation," during malfunctioning and crashes, and after the end of life of the PO.
- *Memorization*. A DT stores all the status changes and events occurred to the PO. A DT represents the status of the PO over time and space.
- *Augmentation*. A DT can extend the PO functions and offer them by means of APIs. Augmentation can add new functionalities that the PO does not support or provide access to data in particular formats.
- *Composability*. Physical systems are aggregations of subsystems and components, a DT must support the correlation of different elementary DTs into complex organizations and provide views on the aggregated DT and individual components.
- *Replication*. A DT can be replicated to serve the needs of different applications. Replicas of the same PO must behave consistently, i.e., they cannot have a different status and they cannot exhibit different behaviors.
- *Accountability/Manageability*. Each DT must be manageable; it is possible to determine its status and activities and to optimize its execution in the framework in which it is operating. It must provide information about the usage of the PO by the applications associated with it. Self-management of DTs is a desirable property for large systems.
- *Servitization*. A DT enables the transformation of a product/artifact into a set of functionalities offered to users. This capability transforms products into software-controlled and connected entities accessible "on-demand" by users.
- *Predictability*. A DT has the ability to simulate its behavior over time. Specific instances of DTs can simulate the behavior of PO in a context at a specific time (in the past or in the future).

Some DT properties need contextualization: replicas are used for tracking POs or for simulate

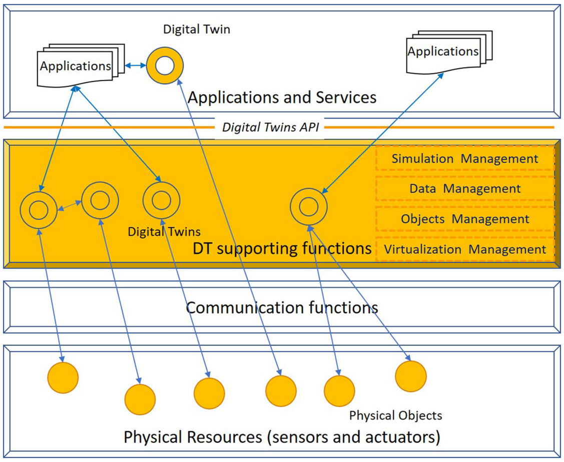

Figure 2. General framework for digital twins.

their behavior. All the replicas of a DT tracking the actual status of a PO should have the same status at the same time *t* (with time limitations explained for entanglement), DTs used for simulations can diverge depending on the simulation conditions.

TOWARDS A GENERAL SOFTWARE FRAMEWORK

A software framework for DTs implements numerous functionalities, e.g., orchestration of objects, cognition for intelligent optimization, self-management to minimize human intervention. Different software architectures may be applicable.[7,8] Some adopt layering principles for organizing functionalities.[9] These architectures must be evaluated to determine how properly they support the DT requirements. Figure 2 represents a generic framework organized in four functional segments. This structure is chosen to illustrate the separation of concerns. IoT devices, communications means (protocols and networks), DT tools and functions will quickly evolve. Layering may prove useful: it is simple enough to be practically used and can allow independent evolution of the different technological layers. Finally, layering is aligned with efforts in standardizing IoT architectures. AIOTI[10] describes an IoT layered architecture that could support the DT approach. However, modularization and linkage of functionalities can prove as practical as layered architectures. Physical resources are devices producing information or acting on the environment (sensors and actuators). The information (commands, events, requests, and status notifications) exchanged between devices and components of the framework is transported by means of communications functions. The DT supporting functions segment represents the execution environment of DTs and the set of intelligent functionalities that DT instances use for orchestrating their interactions and lifecycles (e.g., instantiation, replication, identification, access control, and self-management), managing collected data, and simulation capabilities. This segment mediates the interaction between applications and physical devices. New functions, mechanisms and tools may be progressively designed and added to better meet DTs requirements. The application and services segment supports the applications that use the DTs interfaces and functions. For customization and execution purposes, DT replicas may be locally executed in this environment.

The actual exploitation of the DT as a general-purpose approach occurs if some conditions are met: the availability of a software framework capable of fulfilling the properties of DTs; the ability to create new applications with better functionalities for the user; and the ability to introduce new business models or to improve existing ones. These issues are analyzed in the following sections.

APPLICATION SCENARIOS

DT can be used to implement several scenarios. In this article, two application scenarios, based on properties of DTs and software framework, illustrate possibilities and issues.

Scenario 1: Virtual Object

This example describes how to create the DT of an object. A sensor is chosen, but different (aggregations of) objects may be considered. The needed design tasks are given as follows.

Physical object model: An abstract computable description of the data and behavior of the PO that the DT implements. The sensor status and the measurements it takes in the environment are relevant aspects to consider and to model.

API definition: an interface for requesting the functionalities offered by the DT. Applications use it to get data or to give commands. Data formats and interaction mechanisms should be defined.

Entanglement model: a model governing how PO and DT exchange data and status updates. The framework must offer mechanisms and protocols to support communication. Often the data flow is unidirectional from PO to DT, while it is bidirectional between the DT and applications (data, events, and commands). Commands from the applications to DT are used to govern the DT behavior.

A single PO can be replicated by several DTs. This occurrence accommodates specific requirements of applications: providing special functions, or permit access to the data with different rates, or segmenting and securing the computational space. Replicas can be instantiated in the framework and others in the application space. This option offers flexibility in deployment. Replication of DTs poses interesting issues. When the status of the PO changes, all the replicas need to change status. The propagation of information can be achieved with different mechanisms; PubSub is used in IoT implementations[11] and it suits the case of unidirectional communication (where the PO is a publisher and the replicas are subscribers). If PO status can be modified or it can be instructed (an actuator), the problem of synchronization and consistency of information between the PO and different replicas emerges. This issue can be solved by introducing transaction management functions into the framework.

Scenario 2: Digital Twins in Smart City

Cities are complex systems[12] difficult to control and program. A prerequisite for a smart city is having "measurable" attributes. Sensors and actuators are deployed to capture data that describe its processes. This make a city quantifiable and assessable. Once reliable data are available, artificial intelligence and machine learning, AI/ML, techniques determine patterns and behavioral models of urban systems.[13] The city becomes "understandable," i.e., it is possible to predict events and their associated causes by means of data analysis. DTs are a step toward the programmability of a city: the digital representation of resources makes it possible to design applications that, by interacting with DTs, can affect the behavior of "things" and optimize city resources and their behavior. Several experiences introduce the DT concept in smart cities.[14,15] The DT application introduces new capabilities and has an impact on how to govern the complexity of urban systems. For example, the adoption of the DT approach is improving the usage of building information modeling, e.g., such as given by Delbrügger *et al.*[16]

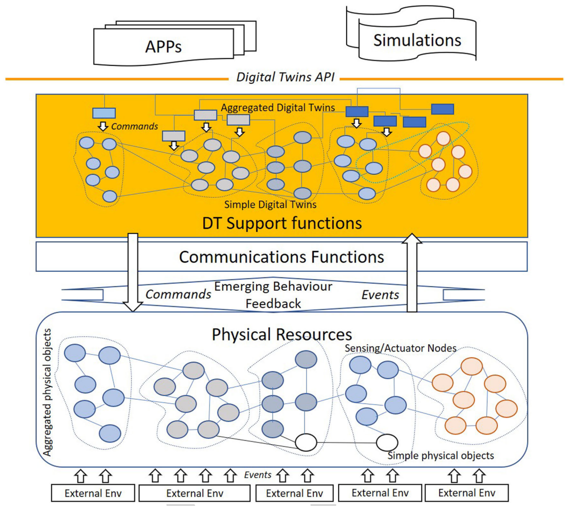

Figure 3. Programmability of complex systems with digital twins.

DTs are programmable objects that can orchestrate, on request of applications, POs like actuators toward the desired behavior. DTs can detect malfunctioning in POs and minimize their impact on other resources. DTs, exploiting advancement in storage technology, can store large datasets and hence supporting the continuous "learning" of city behavior. They offer the possibility to evaluate new policies by simulating future situations. DTs could be utilized to solve a common issue of smart cities: the siloing of application domains (e.g., transportation, energy, and so on). DTs offer interfaces through which different applications can access data, behavior, and status corresponding to specific resources. DT composability enables the creation of object aggregations that support abstract views of larger phenomena. The framework must scale up from few resources to a multitude. Figure 3 represents a DT platform able to capture the behavior of a complex system. Some applications are designed to monitor the situation for applying policies on specific issues (traffic, transportation, etc.). When issues arise, these applications implement mechanisms to minimize impacts on users and optimize resource usage. Other applications are simulations executed to understand how the "smart city" could react to changes in policies of resources' usage. Benefits of applying the DT concept to the "smart city" are many. These advantages depend upon the introduction of higher programmability levels, a strong integration with AI/ML capabilities, the elimination of silo constraints, and the possibility to simulate the functioning of a city.

BUSINESS AS USUAL?

The DT concept has business implications. It changes the nature of POs and products. DT properties facilitate the adoption of servitization, i.e., the ability to transform a physical product into a set of services offered to the user.[17,18] Servitization nurtures customization of functionalities, making products more aligned to the lifestyles and needs of people and it introduces additional innovations in the service area.[19] Products could even be modified, improved, and customized by the user. New functions can be added to products if users' needs change. The customer relationship process could be transformed.[20] Customers are "entangled" with products. If malfunctions emerge in a product, they can be predicted before they actually occur and the impact on users could be mitigated. Maintenance can be improved by focusing more on preventing problems than on solving them. The products' value chain could be largely modified. As the mobile phone industry showed, a rich application layer has a tremendous impact on products and it can determine success in a global market. Customization, and the actors supporting it, could become an important component of the entire value chain of new digital markets.

CONCLUSIONS

The DT approach promises to have impact. It may provide solutions in different sectors (smart cities, e-health, e.g., virtual patient, automotive, cultural heritage), supporting the provision of servitized products, and programmability of complex systems. It fosters new ways of doing business by transforming products into sets of customizable services (an autonomous car and a smart building). A precise definition of the DT can help in better understanding requirements and impacts on IoT software platforms. DTs should be evaluated for their ability to support and deliver the expected benefits. Further investigation on how DTs can enable an ecosystem of applications that are feasible, marketable, and sustainable by the industry is needed. There are two major concerns: feasibility, i.e., the complexity and technical challenges associated with the creation of scalable, secure DT frameworks are such that the construction of a viable platform remains to be proven in-the-large; and user ownership, i.e., the possibility to frame users into a system of products and applications that limit privacy. The DT approach needs validation before it will be widely accepted and used. It already is a powerful means to control, govern, and program the lifecycle of objects, physical resources, and products, for providing better services to end-users. It is a challenge to academia and industry to find relevant applications for it.

REFERENCES

1. M. Grieves, *Product Lifecycle Management: Driving the Next Generation of Lean Thinking*. New York, NY USA: McGraw-Hill, 2006.
2. R. Rosen, G. Von Wichert, G. Lo, and K. D. Bettenhausen, "About the importance of autonomy and digital twins for the future of manufacturing," *IFAC-PapersOnLine*, vol. 48, no. 3, pp. 567–72, Jan. 1, 2015.
3. F. Tao, J. Cheng, Q. Qi, M. Zhang, H. Zhang, and F. Sui, "Digital twin-driven product design, manufacturing and service with big data," *Int. J. Adv. Manuf. Technol.*, vol. 94, no. 9–12, pp. 3563–3576, Feb. 1, 2018.
4. E. Negri, L. Fumagalli, and M. Macchi, "A review of the roles of digital twin in CPS-based production systems," *Procedia Manuf.*, vol. 11, pp. 939–948, 2017.
5. S. Haag and R. Anderl, "Digital twin–proof of concept," *Manuf. Lett.*, vol. 15, pp. 64–66, 2018.
6. C. Steinmetz, A. Rettberg, F. G. C. Ribeiro, G. Schroeder, and C. E. Pereira, "Internet of Things ontology for digital twin in cyber physical systems," in *Proc. VIII Brazilian Symp. Comput. Syst. Eng.*, 2018, pp. 154–159.

7. V. Souza, R. Cruz, W. Silva, S. Lins, and V. Lucena, "A digital twin architecture based on the industrial Internet of Things technologies," in *Proc. IEEE Int. Conf. Consum. Electron.*, 2019, pp. 1–2.
8. S. Malakuti and S. Grüner, "Architectural aspects of digital twins in IIoT systems," in *Proc. 12th Eur. Conf. Softw. Architecture: Companion*, Sep. 24, 2018, pp. 1–2.
9. S. Malakuti, J. Schmitt, M. Platenius-Mohr, S. Grüner, R. Gitzel, and P. Bihani, "A four-layer architecture pattern for constructing and managing digital twins," in *Proc. Eur. Conf. Softw. Architecture*, Sep. 9, 2019, pp. 231–246.
10. AIOTI - WG3 - IoT Standardization "High level architecture," Jun. 2018, Accessed on: Jan. 31, 2020. [Online]. Available: https://aioti.eu/wp-content/uploads/2018/06/AIOTI-HLA-R4.0.7.1-Final.pdf
11. D. Happ, N. Karowski, T. Menzel, V. Handziski, and A. Wolisz, "Meeting IoT platform requirements with open pub-sub solutions," *Ann. Telecommun.*, vol. 72, no. 1/2, pp. 41–52, Feb. 2017.
12. M. Batty, "Complexity in city systems: Understanding, evolution, and design," *A Planner's Encounter With Complexity*. G. de Roo and E. Silva Eds. 2007, pp. 99–122.
13. S. Khan, D. Paul, P. Momtahan, and M. Aloqaily, "Artificial intelligence framework for smart city microgrids: State of the art, challenges, and opportunities," in *Proc. 3rd Int. Conf. Fog Mobile Edge Comput.*, 2018, pp. 283–288.
14. C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Sensing as a service model for smart cities supported by Internet of Things," *Trans. Emerg. Telecommun. Technol.*, vol. 25, no. 1, pp. 81–93, 2014.
15. G. Garfield-Bennett, "Virtual Jersey becomes a reality," *Jersey Evening Post*, Feb. 8, 2019, Accessed on: Jan. 31, 2020. [Online]. Available: https://jerseyeveningpost.com/news/business/2019/02/08/virtual-jersey-becomes-a-reality/
16. T. Delbrügger, L. T. Lenz, D. Losch, and J. Romann, "A navigation framework for digital twins of factories based on building information modeling," in *Proc. 22nd IEEE Int. Conf. Emerg. Technol. Factory Autom.*, 2017, pp. 1–4.
17. C. Kowalkowski, H. Gebauer, B. Kamp, and G. Parry, "Servitization and deservitization: Overview, concepts, and definitions," *Ind. Marketing Manage.*, vol. 60, pp. 4–10, 2017.
18. Q. Qi, F. Tao, Y. Zuo, and D. Zhao, "Digital twin service towards smart manufacturing," *Procedia CIRP*, vol. 72, pp. 237–242, Jan. 1, 2018.
19. P. Zheng, T. J. Lin, C. H. Chen, and X. Xu, "A systematic design approach for service innovation of smart product-service systems," *J. Cleaner Prod.*, vol. 201, pp. 657–667, Nov. 10, 2018.
20. R. N. Bolton *et al.*, "Customer experience challenges: Bringing together digital, physical and social realms," *J. Service Manage.*, vol. 29, pp. 776–808, Oct. 15, 2018.

Roberto Minerva is currently an Associate Professor within the Service Architecture Lab, Institut Mines Telecom–Telecom Sud Paris, Institute Polytechnique de Paris, Paris, France. He received the M.S. degree in computer science from Bari University, Bari, Italy, in 1987, and the Doctoral degree in computer science and telecommunications from UPMC (Paris-Sorbonne University) in 2013. From 1987 to 2016, he was a Researcher then a responsible of the Service Architectures and Network Intelligence area within Telecom Italia Research Center. From 2016 to 2018, he was the Technical Project Leader of SoftFIRE, a European Project devoted to the experimentation of NFV, SDN, and edge computing. He has been the Chairperson of the IEEE IoT Initiative in the period 2014–2016. He is a Senior Member of IEEE. Contact him at roberto.minerva@telecom-sudparis.eu.

Noël Crespi joined the Institut Mines-Telecom SudParis, in 2002, and is currently a Professor and Program Director with the Institut Polytechnique de Paris, leading the Service Architecture Laboratory. He is also an Adjunct Professor with KAIST (South Korea), an Affiliate Professor with Concordia University (Canada), and a Guest Researcher with the University of Goettingen (Germany). His current research interests include data analytics, the Internet of Things, and softwarization. He received the master's degrees from the Universities of Orsay, Paris 11, France, and Kent, Canterbury, U.K., a diplome d'ingénieur from Telecom ParisTech, Paris, France, and the Ph.D. and Habilitation degrees from UPMC (Paris-Sorbonne University). From 1993 to 2002, he worked in the industrial research of telecom organizations in the creation of new services and in standardization. He is a Senior Member of IEEE. Contact him at noel.crespi@mines-telecom.fr.