

Digital Twin Platforms: Requirements, Capabilities, and Future Prospects

Daniel Lehner, Johannes Kepler University Linz

Jérôme Pfeiffer, Erik-Felix Tinsel, and Matthias Milan Strljic, University of Stuttgart

Sabine Sint and Michael Vierhauser, Johannes Kepler University Linz

Andreas Wortmann, University of Stuttgart

Manuel Wimmer, Johannes Kepler University Linz

// Digital twins (DTs) have emerged as a paradigm for the virtual representation of complex systems alongside their underlying hardware. We investigate the benefits of Amazon, Eclipse, and Microsoft DT platforms and assess the extent to which they meet standard requirements. //



Digital Object Identifier 10.1109/MS.2021.3133795
Date of current version: 14 February 2022

A DIGITAL TWIN (DT) is a virtual representation of a system, facilitating bidirectional (Bx) communication between the system and its digital representation.¹ This communication is enabled by explicitly defining the data produced by the system, augmenting it with information about system entities, and realizing “value-adding services” on top of this data-driven definition. Ultimately, DTs are software systems developed by software engineers. Since many commonalities between DTs from different domains exist (see, e.g., Isto et al.² and Barricelli et al.³), creating them from scratch again and again can be deemed inefficient. To meet the need for economical deployment, major software companies have begun providing support for creating and operating DTs, commonly referred to as *DT platforms*. To provide insights for practitioners, the contributions of this article are 1) an elicitation of requirements for DT platforms, 2) a description of the requirements through a use case, and 3) an assessment of the capabilities of selected DT platforms with respect to the requirements.

Instead of a broad overview, which has been previously given,⁴ we aim at analyzing and discussing the capabilities of three example platforms. We further provide an extensive online data set⁵ that can be used and extended by the community to provide evaluations of additional platforms in the future. More specifically, as representatives of platforms offered by commercial cloud providers, we investigate tools provided by Microsoft Azure (AZ) (<https://azure.microsoft.com/services/iot-hub/>, <https://azure.microsoft.com/services/digital-twins/>, and <https://azure.microsoft.com/services/>) and Amazon Web Services (AWS).⁶ As an open source alternative, we investigate the DT solutions part of

the Eclipse (EC) ecosystem (<https://www.eclipse.org/hono/>, <https://www.eclipse.org/vorto/>, and <https://www.eclipse.org/ditto/>).

To ground our discussion, we employ a DT of a “smart room” as a running example throughout this article. Using a measurement system inside the room, indoor air quality properties, such as carbon dioxide levels and

Requirements for DT Platforms

Figure 1 shows the role of DT platforms in the context of ISO23247⁷ for our example. Based on the literature^{2–4, 7–11} and the authors’ expertise with operating and developing DTs in different domains in various multiyear research projects [e.g., the Internet of Production (<https://www.iop.rwth-aachen.de>) and Christian Doppler Laboratory for Model-Integrated Smart Production (<https://cdl-mint.se.jku.at/>)], we derive requirements for the platforms.

R1: Bx Synchronization

Bx synchronization⁸ enables DTs and their physical counterparts to remain in sync. Thus, DTs can receive data from the system and vice versa. Applied to our use case, the DT receives data from individual sensors connected to the Raspberry Pi and stores and analyzes this information. Once a certain air quality threshold is exceeded, the DT sends a command to activate an alarm in a corresponding room.

R2: Convergence

Convergence⁹ is realized between physical and digital spaces, identifying potential differences, and hence it can be adjusted and optimized by physical twins and DTs. Applied to our use

case, the DT should be synchronized with sensors and status LEDs at all times. Should the DT drop out of sync with the physical system, the situation should be detected and rectified.

R3: Verification and Validation

Verification and validation (V&V)⁸ should be provided by the platform, e.g., based on historical data before DTs are deployed, as DTs control critical parts of physical processes. In our use case, before deployment, the DT should be verified against different thresholds and alarms. Furthermore, it should be validated to ascertain whether an acoustic or visible alarm is more suitable for employees in the monitored offices.

R4: Real-Time Behavior

Real-time behavior⁹ for hard constraints between a system and DT on the edge and soft real-time communication with a DT through the cloud need to be taken into consideration. Applied to our use case, actual real-time interaction could be required between the Raspberry Pi and its sensors and LED, and soft real-time communication could be necessary for visualizing the air quality on a dashboard. However, our use case does not rely on hard real-time interaction.

R5: Automation Protocols

Automation protocols¹⁰ are general-purpose procedures of automated systems and directly integrated into the software stack of control apparatus. They are a fundamental requirement to enable interdevice communication and therefore have to be supported. In our case, interaction among different rooms happens via these protocols.

R6: Platform Interoperability

Platform interoperability⁸ involves the extension of an existing DT by

Our findings (see Table 1) show that two-thirds (8/13) of the requirements are fully satisfied by at least one DT platform.

temperature, can be measured by sensors and sent to the system’s DT (e.g., via a Raspberry Pi). If any action is necessary to improve the air quality (e.g., by opening a window), the DT can, for example, activate an alarm on the measurement device. A more detailed description and the prototypical implementation of the measurement system and DT realized on the selected platforms is available at <https://git.io/JDiSH>.

In the following, we 1) derive requirements for DT platforms by instantiating ISO23247⁷ standard for our example (see Figure 1); 2) evaluate the degree to which the selected platforms fulfill these requirements and, based on this, compile a list of practical implications and recommendations for choosing DT platforms for particular projects; and 3) identify future avenues for improving the current capabilities of DT platforms and extending the investigation of the systems in this article.

using value-adding services, such as machine learning, simulation, and visualization.^{4, 12} In our use case, an example of a visualization service

could be a dashboard that shows the development of carbon dioxide values through time for all rooms in a building.

R7: System Interoperability

System interoperability³ enables communication and interaction between DTs of different physical devices.

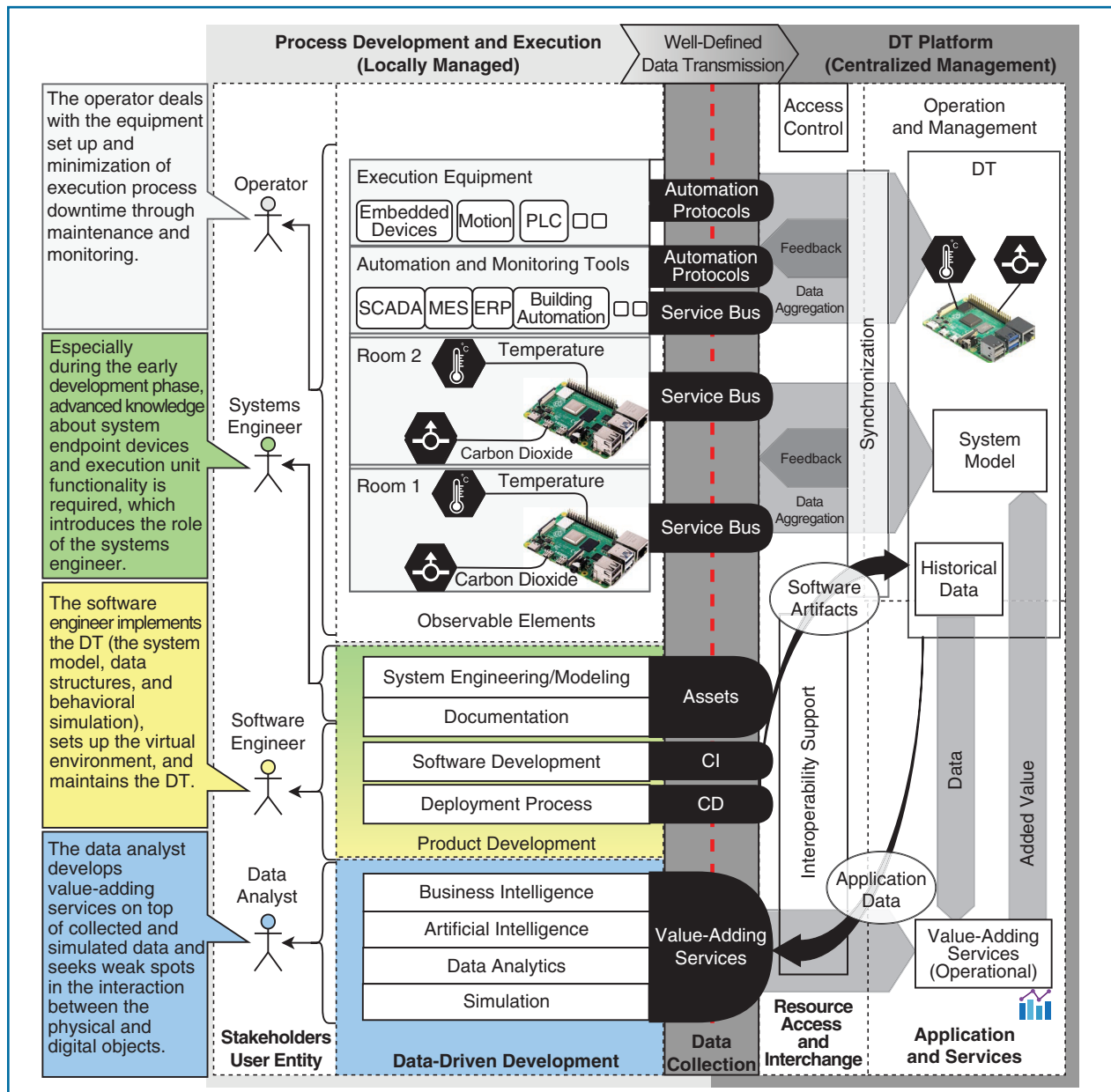


FIGURE 1. An instantiation of ISO23247, *Automation Systems and Integration—Digital Twin Framework for Manufacturing*,⁷ using our running example of the DT of a smart room. Based on this instantiation, we define stakeholders that are concerned with the DT and emphasize the role of the examined DT platforms. Using this figure, we derive requirements for DT platforms and show their relevance through our example. PLC: programmable logic controller; SCADA: supervisory control and data acquisition; CI: continuous integration; CD: continuous deployment; MES: manufacturing execution system; ERP: enterprise resource planning.

Applied to our use case, the DTs of various rooms and even buildings can exchange data, e.g., for recommending that people move to a nearby area with better air quality.

R8: Domain Expert Involvement

Domain expert involvement² means that a platform enables specialists

R10: Modifiability

Modifiability⁸ describes the recurring possibility of changing DTs. As soon as a physical device is changed, its DT must be adapted. The requirement is reflected in our use case by, e.g., replacing the sensor to measure temperature. The new one will report values by using a different physical unit.

Although implementation through the platforms was more efficient than developing DTs from scratch, some functionality had to be manually devised, which may be generalizable, leaving space for future improvements.

to develop/operate a DT without requiring knowledge about the technical realization. For instance, abstraction via modeling languages eases the creation of a DT by offering a structured way to represent physical devices, their types, and associated data.¹³ In our use case, a domain expert may model rooms, controllers, and sensors and specify which data are collected and exchanged.

R9: Connection and Data Security

Connection and data security¹¹ demand protected links and information exchanges and prevent unsafe interactions and unauthorized access. Concerning our use case, for data privacy and to prevent the harmful activation of actuators in offices, we require a secure connection between the DT and Raspberry Pi and stipulate that measured air quality values be safely stored.

R11: Reusability

The reusability⁸ of a DT and its components becomes crucial for software engineers if a company is developing services and if its product line is constantly changing. In such cases, improving the modular character of a DT is a necessity, while reusing single components is an advantage. Applied to our case, we may package the realization of the DT into reusable components that we deploy in other buildings, as well.

R12: Continuous Integration and Continuous Deployment

Continuous integration (CI) and continuous deployment (CD)⁷ include the ongoing incorporation of changes into a DT. After a positive validation by quality assurance, changes are deployed. For our use case, this may be necessary to introduce features, such

as error correction for measured values without stopping the system.

R13: Provisioning

Provisioning⁴ is oriented toward the deployment areas of cloud and edge computing. A platform may operate as a cloud-native as well as an on-premise variant, e.g., to serve time- and security-critical scenarios. For our use case, DTs may be hosted at different places: from edge devices via local servers to remote cloud environments.

Capability Evaluation of DT Platforms

We evaluate the selected platforms regarding their capabilities to cover the aforementioned requirements (We report on the state of the platforms in June 2021. Since they are under continuous development, their capabilities might change in the future.). As a basis for this evaluation, we implemented our use case for each platform. Table 1 summarizes the results and categorizes requirements based on quality characteristics^{14, 15} to provide a structured, standard-based overview, and it aligns every requirement with associated stakeholders introduced in Figure 1. In the following, we summarize the results.

Functional suitability is only partially considered by the examined platforms. Although they all provide basic support for Bx synchronization (R1), implementing it requires a significant amount of manual programming. However, for the convergence (R2) of a system during runtime, no support is provided. Concerning V&V (R3), AWS offers a sophisticated test suite, whereas AZ facilitates basic testing via an integrated CI/CD pipeline. Performance demands support for real-time behavior (R4), which

Table 1. The requirements fulfillment of the AWS-, AZ-, and EC-based DT platforms.

Quality characteristics	Requirements	Metric	Stake-holder	AWS	AZ	EC
Functional suitability represents the degree to which a product or system meets stated and implied functional requirements when used under stipulated conditions.	R1: Bx synchronization ⁸	<ul style="list-style-type: none"> ● DT automatically establishes Bx communication between itself and devices ○ Combination of manual and automatic connection realization ○ Manual establishment of a connection for each device 	SE, OP	●	●	●
	R2: Convergence ⁹	<ul style="list-style-type: none"> ● Platform provides automatic support for ensuring convergence between the DT and its counterpart ○ Platform provides an environment to manually implement convergence ○ No support for convergence 	OP	○	○	○
	R3: V&V ⁸	<ul style="list-style-type: none"> ● Platform provides a test suite for specified functionality, or requirements can be defined and executed before the DT is deployed ○ Invalid structures detected automatically (e.g., sending data from a sensor that does not have a corresponding DT) ○ Platform does not provide any test capabilities 	SE, OP	●	●	○
Performance is related to responsiveness and the efficiency of resource utilization.	R4: Real-time behavior ⁹	<ul style="list-style-type: none"> ● Platform offers a hard real-time interface on the edge and capabilities to measure and evaluate soft real-time constraints in the cloud ○ Platform offers hard real-time interface on the edge only ○ No (hard or soft) real-time support 	OP	●	●	●
Compatibility is the extent to which a system supports the exchange of information with other networks.	R5: Automation protocols ¹⁰	<ul style="list-style-type: none"> ● DT supports at least two widespread protocols, e.g., MTConnect, Open Platform Communications Unified Architecture, Siemens S7, Beckhoff Automation Device Specification, or Rockwell D1 ○ One of these protocols supported ○ No specific support for automation protocols 	OP	●	○	○
	R6: Platform interoperability ⁸	<ul style="list-style-type: none"> ● Platform provides standardized interfaces to 1) retrieve DT data and structural information and 2) make changes to DTs for external services ○ Only one of these options supported ○ None of these options supported 	SE, DA	●	●	●
	R7: System interoperability ³	<ul style="list-style-type: none"> ● Platform provides support for defining connections between different devices via their DTs and for automatically implementing interactions based on these connections ○ Only one of these aspects available ○ None of these aspects available 	SE	●	●	○

(Continued)

Table 1. The requirements fulfillment of the AWS-, AZ-, and EC-based DT platforms. (Continued)

Quality characteristics	Requirements	Metric	Stake-holder	AWS	AZ	EC
Usability describes the extent to which users can employ a system to achieve their goals with effectiveness, and efficiency.	R8: Domain expert involvement ¹³	<ul style="list-style-type: none"> ● Platform enables domain experts to model physical devices and types and data in a structured way and provides a graphical interface ○ Platform provides modeling capabilities without graphical user interface ○ Platform does not support DT modeling 	SE, DE	○	●	●
Security determines the degree to which data and connections are secured such that no unauthorized accesses can occur.	R9: Connection and data security ¹¹	<ul style="list-style-type: none"> ● Platform provides some form of authentication and authorization mechanism when establishing a connection between a DT and physical system and a secure local connection between devices; platform provides capabilities to encrypt data sent to the DT and data stored on the DT itself ○ Platform supports only one of the preceding ○ No support for security 	SE, OP, SysE	●	●	●
Maintainability is the ability to adapt to changes to the environment and requirements of a software product.	R10: Modifiability ⁸	<ul style="list-style-type: none"> ● DT can be modified during runtime ○ DT can be modified while it is inactive after deployment ○ DT cannot be modified after it is deployed 	OP, SE	○	●	●
	R11: Reusability ⁸	<ul style="list-style-type: none"> ● Platform offers custom and prebuilt components that can be reused between projects ○ Components can be reused within project ○ Components not reusable 	SE	●	●	●
Portability describes the efficiency with which hardware and software artifacts can be transferred between two systems.	R12: CI/CD ⁷	<ul style="list-style-type: none"> ● Platform offers complete CI/CD pipeline or integration ○ CI or CD is offered or can be integrated ○ No support or integration 	SE	●	●	○
	R13: Provisioning ⁴	<ul style="list-style-type: none"> ● Platform provides on-site and cloud-native solutions for deployment ○ Only on-site or cloud-native solutions available for deployment ○ Not applicable 	SE, SysE	●	●	○

Evaluation criteria for requirements: ● fulfilled, ○ partially fulfilled, ○ not fulfilled.
SE: software engineer; SysE: systems engineer; OP: operator; DA: data analyst.

is partially covered by all the platforms. Although they all have real-time support for communication on the edge, there is no provision for evaluating (soft) real-time constraints for cloud communication.

Compatibility is partially covered. Automation protocols (R5) are largely neglected, with the exception of AWS, which has support for the Open Platform Communications Unified Architecture. Platform interoperability (R6)

is provided by all the vendors via dedicated interfaces that enable interaction with DTs. While EC does not include support for system interoperability (R7), AZ enables this aspect through the definition of connections between

DTs via relationships in their language. AWS facilitates the implementation of these interdevice connections but does not provide any explicit language support.

Usability plays an important role when domain expert involvement (R8) is required. AZ and EC provide languages for describing DTs and offer graphical tools for creating

DTs based on these languages, while AWS offers no such assistance. Security (R9) is targeted by all examined platforms, as they provide dedicated support for authentication and

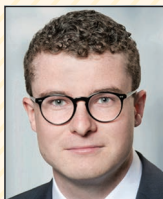
ABOUT THE AUTHORS



DANIEL LEHNER is a Ph.D. candidate in the Department of Business Informatics—Software Engineering and associated with the Christian Doppler Laboratory for Model-Integrated Smart Production, Johannes Kepler University Linz, Linz, 4040, Austria. Lehner received his M.Sc. from TU Wien. His research interests include applying model-driven engineering techniques and practices to digital twins. Contact him at <https://se.jku.at/daniel-lehner/> or daniel.lehner@jku.at.



MATTHIAS MILAN STRLIJIC is a research assistant at the Institute for Control Engineering of Machine Tools and Manufacturing Units, University of Stuttgart, Stuttgart, 70174, Germany. Strljic received his M.Sc. from the University of Stuttgart. His research interests include cloud manufacturing, production process planning, and production automation. Contact him at <https://www.isw.uni-stuttgart.de/en/institute/team/Strljic/> or matthias.strljic@isw.uni-stuttgart.de.



JÉRÔME PFEIFFER is a research assistant at the Institute for Control Engineering of Machine Tools and Manufacturing Units, University of Stuttgart, Stuttgart, 70174, Germany. Pfeiffer received his M.Sc. from RWTH Aachen. Contact him at <https://www.isw.uni-stuttgart.de/en/institute/team/Pfeiffer-00005/> or jerome.pfeiffer@isw.uni-stuttgart.de.



SABINE SINT is a Ph.D. student in the reactive model repositories module, Christian Doppler Laboratory for Model-Integrated Smart Production, Johannes Kepler University Linz, Linz, 4040, Austria. Sint received her M.Sc. from TU Wien. Contact her at <https://se.jku.at/sabine-sint/> or sabine.sint@jku.at.



ERIK-FELIX TINSEL is a research assistant at the Institute for Control Engineering of Machine Tools and Manufacturing Units, University of Stuttgart, Stuttgart, 70174, Germany. Tinsel received his M.Sc. in software engineering from the University of Stuttgart. His research interests include the virtual commissioning and designing a continuous simulation model in the process of plants and machine engineering. Contact him at <https://www.isw.uni-stuttgart.de/en/institute/team/Tinsel/> or erik-felix.tinsel@isw.uni-stuttgart.de.



MICHAEL VIERHAUSER is a postdoctoral researcher in the Secure and Correct Systems Lab, Johannes Kepler University Linz, Linz, 4040, Austria. Vierhauser received his Ph.D. in computer science from Johannes Kepler University Linz. His research interests include safety-critical and cyberphysical systems and runtime monitoring. Contact him at <http://michael.vierhauser.net> or michael.vierhauser@jku.at.



ABOUT THE AUTHORS



ANDREAS WORTMANN is a professor at the Institute for Control Engineering of Machine Tools and Manufacturing Units, University of Stuttgart, Stuttgart, 70174, Germany. Wortmann received his Ph.D. from RWTH Aachen. His research interests include model-driven engineering, software language engineering, and systems engineering, with a focus on Industry 4.0 and digital twins. He is a board member of the European Association for Programming Languages and Systems. Contact him at www.wortmann.ac or wortmann@se-rwth.de.



MANUEL WIMMER is a full professor in, and the head of, the Department of Business Informatics–Software Engineering, Johannes Kepler University Linz, 4040, Linz, Austria. Wimmer received his Ph.D. from TU Wien, Austria. His research interests include software engineering, model-driven engineering, and cyberphysical systems. Contact him at <https://se.jku.at/manuel-wimmer/> or manuel.wimmer@jku.at.

authorization for connections between devices and DTs and other devices and for encrypting data that are sent, received, and stored by a DT.

Maintainability is, to a large extent, covered by all three platforms. Modifiability (R10) is provided by AZ and EC by enabling the adaptation of DTs during runtime via dedicated application programming interfaces, and AWS offers partial modifiability by redeploying software code. Reusability (R11) is covered by AZ via its JavaScript Object Notation-based format to define DTs, by EC via the Vorto repository that facilitates sharing and reusing device specifications, and by AWS via a component feature that enables reusing software code for different devices. Portability is supported by AWS and AZ, as both offer both on-premise and cloud deployment (R13), with additional tooling for setting up a CI/CD pipeline (R12). For EC, there is no dedicated CI/CD support or out-of-the-box support for cloud deployment.

Discussion of Results

Our findings (see Table 1) show that two-thirds (8/13) of the requirements are fully satisfied by at least one DT platform. Even though the majority of the requirements are covered, there is room for improvement to address different stakeholders' demands by providing additional functionality in the future. All investigated platforms perform well with respect to interoperability, security, and reusability, making them highly applicable to safely and efficiently integrating value-adding services with a running system. For AZ and EC, domain expert involvement and modifiability are also strong points. For AWS and AZ, support for CI/CD and provisioning is richer than with EC. In general, AZ and AWS fulfill slightly more requirements than EC. This might be explained by the fact that EC is a collection of tools integrated into a development environment, rather than a full-fledged DT platform. Thus, the missing requirements might be fulfilled by deploying DTs developed with EC to cloud platforms offering additional features.

Convergence and automation protocols are mostly neglected by all the platforms even though they are essential requirements for DTs. For convergence, providing a generalized solution within a platform also seems challenging in the current state of practice, as today's convergence tools depend on the type of system that DTs are developed for. Thus, future research is required for how to identify commonalities to enable a general convergence framework. To provide further support, dedicated servers offering automation protocols may be integrated into the platforms and adapters (e.g., based on HTTP that is already well supported).


In conclusion, we can state that current DT platform capabilities satisfy a good number of the desired requirements, providing many benefits to practitioners. For our example use case, we were able to implement the smart room DT on all platforms. Although implementation through the platforms was more efficient than developing DTs from scratch, some functionality had to be manually devised, which may be

generalizable, leaving space for future improvements.

Based on the derived requirements and discussion of the current capabilities of selected DT platforms, we identify the following avenues for future work:

1. To improve interoperability and reusability, instead of providing proprietary DT languages, standardized languages, e.g., those for cyberphysical systems, may be employed. This would facilitate combining and switching among platforms to provide unified support for existing automation protocols and store DT definitions in a common repository independent from employed systems.
2. The dynamic composition of different DTs to build systems and even systems of systems from individual components in a bottom-up manner could be enabled by providing rich interface descriptions.
3. Interfaces as well as command models may also be required for developing value-adding services based on simulation and machine learning and their integration with DTs.

Finally, we provide future prospects for the presented evaluation framework. To extend the investigation of existing platforms, we plan to 1) implement different value-added services based on the DT platforms for the given use case, 2) extend our evaluation catalogue with additional requirements, e.g., to account for the ease of implementation of value-added services, and 3) assess additional DT

platforms. For this, we provide a living document⁵ and invite the whole community to participate. 

Acknowledgments

The work was supported by the Austrian Federal Ministry for Digital and Economic Affairs; National Foundation for Research, Technology, and Development; and Linz Institute of Technology (grant LIT-2019-7-INC-316).

References

1. W. Kritzinger, M. Karner, G. Traar, J. Henjes, and W. Sihn, "Digital twin in manufacturing: A categorical literature review and classification," *IFAC-PapersOnLine*, vol. 51, no. 11, pp. 1016–1022, 2018, doi: 10.1016/j.ifacol.2018.08.474.
2. B. R. Barricelli, E. Casiraghi, and D. Fogli, "A survey on digital twin: Definitions, characteristics, applications, and design implications," *IEEE Access*, vol. 7, pp. 167,653–167,671, Nov. 2019, doi: 10.1109/ACCESS.2019.2953499.
3. J. C. Kirchhof, J. Michael, B. Rumpe, S. Varga, and A. Wortmann, "Model-driven digital twin construction: Synthesizing the integration of cyber-physical systems with their information systems," in *Proc. MODELS*, 2020, pp. 90–101.
4. Q. Qi *et al.*, "Enabling technologies and tools for digital twin," *J. Manuf. Syst.*, vol. 58, pp. 3–21, Jan. 2021, doi: 10.1016/j.jmsy.2019.10.001.
5. D. Lehner, "Digital twin platforms: Current capabilities and future prospects," GitHub. https://github.com/derlehner/DT_Platform_Comparison (accessed: Dec. 14, 2021).
6. "What is AWS IoT Greengrass?" Amazon Web Services. <https://docs.aws.amazon.com/greengrass/v2/developerguide/what-is-iot-greengrass.html> (accessed: Dec. 14, 2021).
7. *Automation Systems and Integration-Digital Twin Framework for Manufacturing—Part 1: Overview and general principles*, 2020. [Online]. Available: <https://www.iso.org/standard/75066.html>
8. J. Moyne *et al.*, "A requirements driven digital twin framework: Specification and opportunities," *IEEE Access*, vol. 8, pp. 107,781–107,801, 2020, doi: 10.1109/ACCESS.2020.3000437.
9. L. F. C. Durão, S. Haag, R. Anderl, K. Schützer, and E. Zancul, "Digital twin requirements in the context of Industry 4.0," in *Proc. PLM*, 2018, pp. 204–214.
10. K. Y. H. Lim, P. Zheng, and C.-H. Chen, "A state-of-the-art survey of digital twin: Techniques, engineering product lifecycle management and business innovation perspectives," *J. Intell. Manuf.*, vol. 31, no. 6, pp. 1313–1337, 2020, doi: 10.1007/s10845-019-01512-w.
11. A. Redelinghuys, A. H. Basson, and K. Kruger, "A six-layer architecture for the digital twin: A manufacturing case study implementation," *J. Intell. Manuf.*, vol. 31, no. 6, pp. 1383–1402, 2020, doi: 10.1007/s10845-019-01516-6.
12. D. Jones, C. Snider, A. Nassehi, J. Yon, and B. Hicks, "Characterising the Digital Twin: A systematic literature review," *CIRP J. Manuf. Sci. Technol.*, vol. 29, pp. 36–52, May 2020, doi: 10.1016/j.cirpj.2020.02.002.
13. A. Wortmann, O. Barais, B. Combe-male, and M. Wimmer, "Modeling languages in Industry 4.0: An extended systematic mapping study," *Softw. Syst. Model.*, vol. 19, no. 1, pp. 67–94, 2020, doi: 10.1007/s10270-019-00757-6.
14. L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, 3rd ed. Reading, MA: Addison-Wesley, 2012.
15. *System and Software Quality Models*, ISO/IEC 25010, 2010.