Digital Twin Platforms: Requirements, Capabilities, and Future Prospects

Daniel Lehner*, Jérôme Pfeiffer†, Erik-Felix Tinsel†, Matthias Milan Strljic†, Sabine Sint*, Michael Vierhauser‡, Andreas Wortmann†, Manuel Wimmer*
* Christian Doppler Laboratory for Model-Integrated Smart Production (CDL-MINT)
Institute for Business Informatics - Software Engineering
Johannes Kepler University Linz, Linz, Austria
† Institute for Control Engineering of Machine Tools and Manufacturing Units
University of Stuttgart, Stuttgart, Germany
‡ LIT Secure and Correct Systems Lab
Johannes Kepler University Linz, Linz, Austria

***Abstract*—Cyber-Physical Systems are increasingly ubiquitous in our society. With complexity and interconnection, we need new concepts, methods, and tools to systematically leverage data obtained from these systems to better understand, control, and optimize their behavior. Digital Twins have emerged as a new paradigm for the virtual representation of complex systems alongside their underlying hardware. With these systems being connected to another and related systems, major software companies have brought forth platforms for the development and operations of Digital Twins, offering different techniques and capabilities to represent a physical system. We investigate the benefits of three selected Digital Twin platforms. To this end, we (i) collected requirements for Digital Twins based on ISO/DIS 23247 and authors' experience, and back them with existing literature, and (ii) assess to which extent the selected Digital Twin platforms of Amazon, Eclipse, and Microsoft cover these requirements.**

A Digital Twin (DT) is a virtual representation of a system, facilitating bi-directional 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 the system entities, and realizing "value-adding services" on top of this data-driven definition of the system.

Ultimately, DTs are software systems developed by software engineers. As many commonalities between DTs from different domains exist (cf. e.g., [2, 3]), creating them from scratch again and again can be deemed inefficient. To account for the need for efficient development of DTs, major software companies have started providing support for developing and operating DTs, commonly referred to as DT platforms. To provide insights for practitioners, the contribution of this paper is (i) an elicitation of requirements for such DT platforms, (ii) a description of these requirements by a common use case, and (iii) an assessment of current capabilities of selected DT platforms with respect to these requirements.

Instead of aiming for a broad overview, which has been given previously [4], we aim at analyzing and discussing capabilities of three example platforms for practitioners. We further provide an extensive online dataset[1] that can be used and extended by the community to provide evaluations of additional platforms in the future. More specifically, as representatives for platforms offered by commercial cloud providers, we investigate tools provided by Microsoft Azure (AZ)[2] and Amazon Web Services (AWS)[3]. As an open-source alternative, we investigate the DT solutions part of the Eclipse (EC) ecosystem[4].

In order to ground our discussion, we use the DT of a "smart room" as a running example throughout this paper. Using an air quality measurement system inside a room, indoor air quality properties such as CO_2 or temperature can be measured by sensors and sent to its DT (e.g., via a Raspberry Pi). If any action is necessary to improve air quality (e.g., by opening a window), the DT can, for example, activate an alarm on the measurement device. A more detailed description together with the prototypical implementation of the measurement system and the DTs realized in the selected platforms is available online[5].

In the following, we (i) derive requirements for DT platforms by instantiating the ISO/DIS 23247 [5] standard for the running example (cf. Fig. 1), (ii) evaluate to which degree the selected platforms fulfill these requirements, and, based on this, compile a list of practical implications and recommendations for choosing DT platforms in particular projects, and (iii) derive future prospects for improving the current capabilities of DT platforms and extending the investigation of DT platforms started in this paper.

Requirements for DT Platforms

Fig. 1 shows the role of DT platforms in the context of ISO/DIS 23247 [5] for our running example. Based on existing literature [2, 3, 4, 5, 6, 7, 8, 9] and the authors' expertise in

[1] https://github.com/derlehner/DT_Platform_Comparison

[2] (IoTHub: https://azure.microsoft.com/services/iot-hub/, DT service: https://azure.microsoft.com/services/digital-twins/, and TSI database: https://azure.microsoft.com/services)

[3] AWS IoT Greengrass https://docs.aws.amazon.com/greengrass/v2/developerguide/what-is-iot-greengrass.html

[4] Eclipse Hono: https://www.eclipse.org/hono/, Vorto: https://www.eclipse.org/vorto/, and Ditto: https://www.eclipse.org/ditto/

[5] https://github.com/derlehner/IndoorAirQuality_DigitalTwin_Exemplar

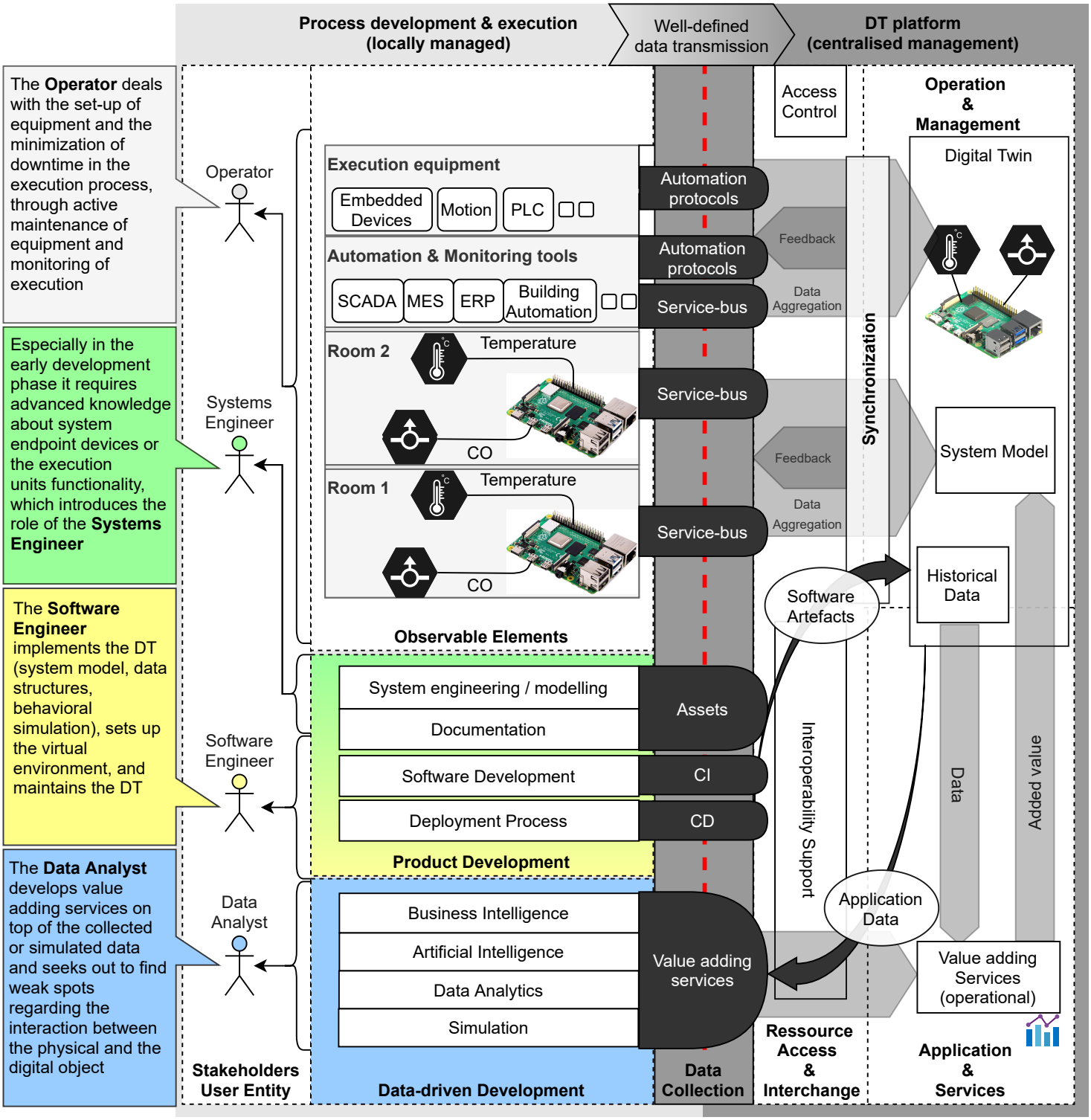


Fig. 1. Instantiation of the standard "ISO/DIS 23247 automation systems and integration-digital twin framework for manufacturing" [5] using our running example of the DT of a smart room. Based on this instantiation, we define stakeholders that are concerned by the DT, and emphasize the role of the examined DT platforms. Based on this figure, we derive requirements for DT platforms, and show their relevance using our DT example.

operating and developing DTs in different domains, in various multi-year research projects (e.g., IoP[6] and CDL-MINT[7]), we derive requirements for such DT platforms.

[6]https://www.iop.rwth-aachen.de
[7]https://cdl-mint.se.jku.at

R1: Bi-Directional (Bx) Synchronization [6] enables DTs and their physical counterparts to keep 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 Pis and stores and analyzes this data. Once a certain air quality threshold is exceeded, the DT sends a command to activate the alarm in the corresponding room.

R2: Convergence [7] is realized between physical and digital spaces, identifying potential differences, and can hence, be optimized and adjusted by physical and digital twins. Applied to our use case, the DT should be synchronized with sensors and status LEDs at all times. Should, at some point, the DT become out of sync with the physical system, this should be detected and repaired.

R3: Verification and Validation (V&V) [6] 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 whether an acoustic or visible alarm is more suitable for employees in the monitored offices.

R4: Real-time behavior [7] for hard real-time constraints between the system and the DT on the edge, as well as soft real-time for communicating with a DT over the cloud needs 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 for visualizing current air quality in a dashboard. However, our use case does not rely on hard real-time interaction.

R5: Automation Protocols [8] are general-purpose protocols of automated systems and are directly integrated into the software stack of control systems. They are a fundamental requirement to enable inter-device communication, and therefore have to be supported. In our case, the interaction between different rooms happens via such automation protocols.

R6: Platform Interoperability [6] involves the extension of an existing DT platform using value-adding services such as machine learning, simulation, or visualization [4, 10] In our use case, an example of a visualization service could be a dashboard that shows the development of CO_2 values over time for all rooms in a building.

R7: System Interoperability [3] enables communication and interaction between DTs of different physical devices. Applied to our use case, the DTs of different rooms, or even buildings can exchange data, e.g., for recommending to move to a nearby room with better air quality.

R8: Domain Expert Involvement [2] means that the platform enables domain experts 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 [11]. In our use case, a domain expert may model rooms, controllers, and sensors and specify the data that is collected and exchanged.

R9: Connection and Data Security [9] demands a secure connection and data exchange, and prevents any harmful connections and unauthorized access. Concerning our use case, for data privacy, and to prevent harmful activation of actuators in offices, we require a secure connection between DTs and Raspberry Pis, and that the measured air quality values are stored securely.

R10: Modifiability [6] describes the recurring possibility to change DTs. As soon as a physical device is changed, the DT must be adapted. The requirement is reflected in our use case by, e.g., replacing the sensor to measure the temperature. The new sensor will report the measured values using a different physical unit.

R11: Reusability [6] of a DT or its components becomes crucial for software engineers if the company is either developing services or if their product line is under constant change. In such cases, improving the modular character of 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 and reuse them for other buildings as well.

R12: Continuous Integration & Continuous Deployment (CI/CD) [5] includes the continuous integration of changes into the DT. After a positive validation by quality assurance changes are deployed. For our use case, this may be necessary to introduce new features such as error correction for measured values without stopping the running system.

R13: Provisioning [4] is oriented towards 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-critical or security-critical scenarios. For our use case, the DTs may be hosted at different places: from edge devices over local servers to remote cloud environments.

Capability Evaluation of DT Platforms

We evaluate the selected platforms regarding their capabilities to cover the aforementioned requirements. As a basis for this evaluation, we implemented our use case for each platform[8].

Table I summarizes the results of this evaluation and categorizes the requirements based on quality characteristics [12, 13] to provide a structured, standard-based overview, as well as aligns each requirement with associated stakeholders as introduced by Fig. 1. In the following, we summarize the results by giving an overview of the current capabilities of the examined platforms.

Functional Suitability is only partially considered by the examined platforms. Although all platforms provide basic support for bi-directional synchronization (R1), implementing this synchronization requires a significant amount of manual programming effort. However, for the convergence (R2) of

[8]We report on the state of the platforms in June 2021. As the platforms are under continuous development, their capabilities might change in the future.

a system during runtime, no support is provided so far. Concerning V&V (R3), AWS offers a sophisticated test suite, whereas AZ allows for basic testing via the integrated CI/CD pipeline.

Performance demands support for real-time behavior (R4), which is partially covered by all platforms. Although they all provide real-time support for communication on the edge, there is no support for evaluating (soft) real-time constraints for cloud communication.

Compatibility is partially covered. Automation protocols (R5) are still largely neglected, with the exception of AWS, offering support for OPC-UA. Platform interoperability (R6) is provided by all vendors via dedicated interfaces that allow interaction with DTs. While EC does not provide support for system interoperability (R7), AZ enables this aspect through the definition of connections between DTs via relationships in their DT language. AWS facilitates the implementation of these inter-device connections, but does not provide any explicit language support.

Usability plays an important role when domain expert involvement (R8) is required. AZ and EC both provide languages for describing DTs in their platform and offer graphical tools for creating DTs based on these languages, while AWS offers no such support.

Security (R9) is targeted by all examined platforms, as they provide dedicated support for authentication and authorization for connections between devices and DTs, or other devices, and for encrypting data that is sent, received, or stored by a DT.

Maintainability is, to a large extent, covered by all three platforms. Modifiability (R10) is covered by AZ and EC by enabling the adaptation of DTs during runtime via dedicated APIs, and AWS offers partial modifiability by redeploying software code. Reusability (R11) is covered by AZ via their JSON-based format to define DTs, by EC via the Vorto repository that allows to share and re-use existing device specifications, and in AWS via the component feature that allows reusing existing software code for different devices.

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

Convergence and automation protocols are currently mostly neglected by all investigated platforms, although being an essential requirement for DTs. For convergence, providing a generalized solution within a platform also seems challenging in the current state-of-practice, as current convergence solutions depend on the type of system that DTs are developed for. Thus, future research is required on how to identify commonalities to enable a general convergence framework. To provide further support for automation protocols, dedicated servers offering such protocols may be integrated in the platforms or adapters (e.g., based on HTTP that is already well-supported) to these automation protocols may be developed.

As a conclusion, it can be stated that current capabilities of DT platforms already cover a good amount of desired requirements, providing many benefits to practitioners implementing DTs. For our example DT use case, we were able to implement the DT of the smart room in all platforms. Although the implementation using the platform was more efficient than developing DTs from scratch, some functionality had to be manually developed which may be generalizable, leaving also space for future improvements of the platforms.

FUTURE PROSPECTS

Based on the derived requirements and discussion of current capabilities of selected DT platforms, we identify the following future prospects for such platforms: (i) to improve interoperability and reusability, instead of providing proprietary DT languages, standardized languages, e.g., already standardized languages for Cyber-Physical Systems, may be employed. This would allow to combine and to switch between different platforms, to provide unified support for existing automation protocols, and to store DT definitions in a common repository independent from the employed platforms; (ii) to enable the dynamic composition of different DTs to build systems, or even system-of-systems, from individual components in a bottom-up manner by providing rich interface descriptions; (iii) such interfaces, as well as command models may be also required for developing value-adding services based on simulation or 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 (i) implement different value-added services based on the existing DT platforms for the given use case, (ii) extend our evaluation catalog with additional requirements, e.g., to account for the ease of implementation for value-added services, and (iii) to evaluate additional DT platforms. For this, we provide a living document[1] and invite the whole community to participate in this process.

TABLE I
THE FULFILLMENT OF THE FOUND REQUIREMENTS IS COMPARED BETWEEN THE AWS, AZURE AND ECLIPSE BASED DT PLATFORMS.
EVALUATION CRITERIA FOR REQUIREMENTS: ● = FULFILLED, ◑ = PARTIALLY FULFILLED, ○ = NOT FULFILLED
ABBREVIATIONS FOR STAKEHOLDERS: SE = SOFTWARE ENGINEER, SYSE = SYSTEMS ENGINEER, OP = OPERATOR, DA = DATA ANALYST
ABBREVIATIONS FOR DT PLATFORMS: AWS = DT PLATFORM BY AMAZON WEB SERVICES, AZ = DT PLATFORM BY MICROSOFT AZURE, EC = DT PLATFORM BY ECLIPSE,

Quality Characteristics	Requirements	Metric	Stake-holder	AWS	AZ	EC
Functional Suitability represents the degree to which a product or system meets the stated and implied functional requirements when used under specified conditions	R1: Bx Synchronization [6]	● DT automatically establishes a bi-directional communication between DT and devices. ◑ Combination of manual and automatic connection realization. ○ Manual establishment of a connection for each device.	SE, OP	◑	◑	◑
	R2: Convergence [7]	● Platform provides automatic support for ensuring convergence between the DT and its counterpart. ◑ Provides a environment to manually implement convergence. ○ No support for convergence.	OP	○	○	○
	R3: V&V [6]	● Platform provides a test suite for specified functionality or requirements can be defined and executed before the DT is deployed. ◑ Invalid structures are 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 [7]	● 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 a hard real-time interface on the edge only. ○ No (hard or soft) real-time support at all.	OP	◑	◑	◑
Compatibility regards the extent to which a system supports the exchange of information with other systems	R5: Automation Protocols [8]	● DT supports at least 2 widespread protocols, e.g., MTConnect, OPC UA, Siemens S7, Beckhoff ADS, and Rockwell D1. ◑ One of these protocols is supported. ○ No specific support for automation protocols.	OP	◑	○	○
	R6: Platform Interoperability [6]	● Platform provides standardized interfaces to (i) retrieve data and structural information of DTs and (ii) make changes to the DTs for external services. ◑ Only one of these options is supported. ○ None of these options is supported.	SE, DA	●	●	●
	R7: System Interoperability [3]	● 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 is available. ○ None of these aspects is available.	SE	◑	◑	○
Usability describes to which extent users can use a system to achieve their goals with effectiveness, and efficiency	R8: Domain Expert Involvement [11]	● Platform enables domain experts to model physical devices, their types, and data in a structured way and provides a graphical interface for this. ◑ Platform provides modelling capabilities without graphical user interface. ○ Platform does not support DT modelling.	SE, DE	○	●	●
Security determines the degree to which data and connections are secured, such that no unauthorized accesses can occur	R9: Connection & Data Security [9]	● Platform provides some form of authentication and authorization mechanisms when establishing a connection between the DT and physical system and secure local connection between devices. In addition, it provides capabilities to encrypt both the data that is sent to the DT, and data stored on the DT itself. ◑ It only supports one of the above. ○ No support for security at all.	SE, OP, SysE	●	●	●
Maintainability describes the ability to adapt to changes, regarding the environment and the requirements of a software product	R10: Modifiability [6]	● DT can be modified during runtime. ◑ DT can be modified while the DT is inactive after it has been deployed. ○ DT cannot be modified after it has been deployed.	OP, SE	◑	●	●
	R11: Reusability [6]	● Platform offers custom and pre-built components that can be reused between projects. ◑ Components can be reused within a project. ○ Components cannot be reused at all.	SE	●	●	●
Portability describes the efficiency with which hardware or software artifacts can be transferred between two systems	R12: CI/CD [5]	● Platform offers a complete CI/CD pipeline or integration ◑ CI or CD is offered or integratable ○ No support or integration is offered	SE	●	●	○
	R13: Provisioning [4]	● Platform provides on-premise & cloud-native solutions for deployment ◑ Only on-premise or cloud-native solutions are available for deployment ○ Not applicable	SE, SysE	●	●	◑

ACKNOWLEDGMENTS

The work has been supported by the Austrian Federal Ministry for Digital and Economic Affairs and the National Foundation for Research, Technology and Development (CDG), and the Linz Institute of Technology (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.

[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, 2019.

[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. of MODELS*, 2020, pp. 90–101.

[4] Q. Qi, F. Tao, T. Hu, N. Anwer, A. Liu, Y. Wei, L. Wang, and A. Nee, "Enabling technologies and tools for digital twin," *Journal of Manufacturing Systems*, vol. 58, pp. 3–21, 2021.

[5] ISO/DIS, "ISO/DIS 23247 automation systems and integration-digital twin framework for manufacturing," International Standardization Organization (ISO), Tech. Rep., 2020.

[6] J. Moyne, Y. Qamsane, E. C. Balta, I. Kovalenko, J. Faris, K. Barton, and D. M. Tilbury, "A requirements driven digital twin framework: Specification and opportunities," *IEEE Access*, vol. 8, pp. 107 781–107 801, 2020.

[7] 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. of PLM*. Springer, 2018, pp. 204–214.

[8] 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," *Journal of Intelligent Manufacturing*, vol. 31, p. 1313–1337, 2020.

[9] A. Redelinghuys, A. H. Basson, and K. Kruger, "A six-layer architecture for the digital twin: a manufacturing case study implementation," *Journal of Intelligent Manufacturing*, vol. 31, p. 1383–1402, 2020.

[10] D. Jones, C. Snider, A. Nassehi, J. Yon, and B. Hicks, "Characterising the Digital Twin: A systematic literature review," *CIRP Journal of Manufacturing Science and Technology*, vol. 29, pp. 36–52, 2020.

[11] A. Wortmann, O. Barais, B. Combemale, and M. Wimmer, "Modeling languages in Industry 4.0: an extended systematic mapping study," *Software and Systems Modeling*, vol. 19, no. 1, pp. 67–94, 2020.

[12] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, 3rd ed. Addison-Wesley Professional, 2012.

[13] ISO/IEC, "ISO/IEC 25010 system and software quality models," International Standardization Organization (ISO), Tech. Rep., 2010.

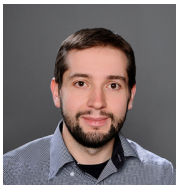
Daniel Lehner is a PhD candidate at the Department of Business Informatics - Software Engineering, and also associated with the Christian Doppler Laboratory for Model-Integrated Smart Production (CDL-MINT), both at Johannes Kepler University Linz. His research interests include applying Model-Driven Engineering techniques and practices to Digital Twins. Contact him at daniel.lehner@jku.at Find out more about this author, including contact details on his website: https://se.jku.at/daniel-lehner/

Jérôme Pfeiffer is a research assistant at the Institute for Control Engineering of Machine Tools and Manufacturing Units (ISW) of the University of Stuttgart. Find out more about this author, including contact details on his website: https://www.isw.uni-stuttgart.de/en/institute/team/Pfeiffer-00005/

Erik-Felix Tinsel is a research assistant at the Institute for Control Engineering of Machine Tools and Manufacturing Units (ISW) of the University of Stuttgart. Find out more about this author, including contact details on his website: https://www.isw.uni-stuttgart.de/en/institute/team/Tinsel/

Matthias Milan Strljic is a research assistant at the Institute for Control Engineering of Machine Tools and Manufacturing Units (ISW) of the University of Stuttgart. Find out more about this author, including contact details on his website: https://www.isw.uni-stuttgart.de/en/institute/team/Strljic/

Sabine Sint is currently working as PhD student in the Christian Doppler Laboratory for Model-Integrated Smart Production (CDL-MINT) at the JKU Linz in the module Reactive Model Repositories. Find out more about this author, including contact details on her website: https://se.jku.at/sabine-sint/



Michael Vierhauser is a post-doctoral researcher at the Secure and Correct Systems Lab at the Johannes Kepler University Linz, Austria. He holds a Master's degree in Software Engineering and Ph.D. in Computer Science from the Johannes Kepler University Linz. His current research interests include safety-critical and cyber-physical systems, and runtime monitoring. Further information about him can be found at http://michael.vierhauser.net. Contact him at michael.vierhauser@jku.at

Andreas Wortmann is a professor at the Institute for Control Engineering of Machine Tools and Manufacturing Units (ISW) of the University of Stuttgart where he conducts research on model-driven engineering, software language engineering, and systems engineering with a focus on Industry 4.0 and digital twins. Find out more at www.wortmann.ac or https://www.isw.uni-stuttgart.de/en/

Manuel Wimmer is Full Professor and Head of the Department of Business Informatics – Software Engineering at Johannes Kepler University Linz. His research interests include Software Engineering, Model-Driven Engineering, and Cyber-Physical Systems. Find out more at https://se.jku.at/manuel-wimmer/