Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 119 (2023) 867–872

www.elsevier.com/locate/procedia

33rd CIRP Design Conference 2023

Self-adaptive digital twin reference architecture to improve process quality

Ann-Kathrin Splettstößer[a,*], Carsten Ellwein[a], Andreas Wortmann[a]

[a] *Institute for Control Engineering of Machine Tools and Manufacturing Units, University of Stuttgart, Seidenstr. 36, D-70174 Stuttgart, Germany*

* Corresponding author. Tel.: +49-711-685-81175; *E-mail address:* ann-kathrin.splettstoesser@isw.uni-stuttgart.de

Abstract

The aim of software-defined manufacturing is a more adaptable production through software. This requires highly adaptive, versatile cyber-physical production systems (CPPSs), which react flexibly to changes and perform crucial adjustments. Self-adaptive digital twins (DTs) enable adaptability through monitoring the behavior of a CPPS, identifying problems, and determining adjustments for the optimization of processes. DTs work with an idealized representation of the CPPS. Changes in system behavior that negatively impact the process execution, e.g., caused by tool wear or variations of environmental influences, and, thus, the quality of the resulting product, are ignored. This ignorance leads to defective products, sub-optimal product quality, and hazardous CPPS states. To mitigate this, we present an approach to incorporate quality awareness into a model-driven reference architecture of a self-adaptive DT. The devised concept automatically analyzes and optimizes system behavior. To this end, a component for quality evaluation extends the reference architecture. The DT and CPPS are connected via OPC UA. To validate the approach, a fiber laser machine is utilized. The DT can operate both on a physical and a virtual laser machine.

Keywords: Digital Twin; Model-Driven Development; Software-Defined Manufacturing; Quality Awareness

1. Introduction

In industrial manufacturing, traditional production processes are challenged by the transition away from mass production towards mass customization [1]. Digital technologies are integrated to respond to these developments [1]. Through connecting the physical and the virtual world, greater efficiency and accuracy, as well as a higher level of productivity, can be achieved [1, 2]. Digital twins (DTs) can handle, analyze, and evaluate large amounts of data generated during production [3]. A DT virtually represents a cyber-physical production system (CPPS) using comprehensive data and models [4]. CPPS and DT are interconnected and influence each other [3, 4]. Realized as a self-adaptive system [5], the DT is capable of autonomously responding to system inconsistencies and attuning itself to the system's environment and requirements [6]. Using DTs opens up new opportunities for system data processing, e.g., simulation for process optimization or generation of system behavior predictions [4]. The manual implementation of a DT is challenging and time-consuming [6]. Model-driven development [7] of DTs uses several models with different levels of abstraction to derive and generate the necessary source code [6].

This paper addresses how to modify the reference architecture of a self-adaptive DT to reflect and compensate for changes in the performance and environment of a CPPS. Such a quality-aware DT can accurately map highly individual machine capabilities and machine behaviors, which results in the improvement of process quality. The term "process quality" denotes the quality of a manufacturing process to obtain a consistent product. To achieve a quality-aware DT, the key contributions are:

a. Including a quality assessor component to a model-driven, self-adaptive DT reference architecture [6],
b. Establishing data exchange between DT and CPPS via open platform communications unified architecture (OPC UA),
c. Enhancing system testability and trainability by connecting the physical CPPS or its virtual counterpart to the DT.

10.1016/j.procir.2023.03.131

In the following, Section 2 introduces preliminaries and highlights related research. Subsequently, Section 3 depicts a modified DT reference architecture with OPC UA communication for improving process quality. Section 4 presents an application example for a laser machine. Section 5 discusses the results. Finally, Section 6 concludes by summarizing the work and providing an outlook.

2. Preliminaries and Related Work

The approach to developing a quality-aware DT builds upon a variety of concepts. This section introduces the preliminaries and discusses related research. It concludes by examining the research gap.

2.1. Digital Shadow and Digital Twin

Although the use of DTs is widespread in many application areas [8], separate DT definitions exist for different fields, e.g., for product design [9] or smart manufacturing [10]. No universally accepted definition of the term DT exists [11]. A systematic mapping study [8] investigates 1471 unique publications about DTs to determine the nature of DTs. [12] investigates three commercially available DTs concerning their capabilities and expandability.

The DT concept commonly describes a virtual counterpart of a physical object [3]. We follow the differentiation between digital shadow (DS) and DT, according to the integration of automated data flow between the physical and the virtual object. The DS enables one-way automated data flow from a physical object to its virtual representation [3]. Changes in the state of the physical object directly influence the corresponding digital representation [13]. Integrating a bidirectional data flow between physical and virtual object yields a DT [3]. Variations in the physical object state directly lead to an adjustment of the virtual object. Inversely the same applies, changes in the state of the virtual object directly influence the physical object. This enables the virtual representation to function as a control instance of the physical object. A DT system encompasses the whole structure, consisting of the virtual and physical object, and the infrastructure for data exchange, synchronization, and services.

2.2. Self-adaptive Software System

Distributed, mobile, and integrated software systems of increasing size and intricacy complicate manual monitoring and control [14]. These systems operate in a highly dynamic environment, which requires flexible reactions to changing conditions. Self-adaption enables responding to alterations by adjusting system behavior at runtime.

One method to implement self-adaptation is the monitor-analyze-plan-execute over a shared knowledge base (MAPE-K) [15] control and feedback loop, which follows a sequence of the four eponymous phases to adapt the system behavior of a CPPS [16]. The CPPS interacts via sensors and actuators with the MAPE-K loop [17]. During the monitoring phase, data is collected, filtered, and summarized. The analysis phase provides a function to correlate and model complex situations. Thus, the MAPE-K system becomes familiar with its operating environment and is able to predict future situations. In order to specify measures needed to achieve the stated goals and objectives, the planning phase utilizes available mechanisms which take framework conditions into account. The task of the execution phase is to supervise the realization of the measure, determined in the previous phase, on the CPPS under constant consideration of dynamic updating.

A technical realization of the analysis phase is event-condition-goal (ECG) models [18]. ECG models determine system states. In ECG, the occurrence of an event satisfying a predefined condition triggers the automatic output of a goal objective. The planning phase may utilize case-based-reasoning (CBR) to plan compensating measures [19]. CBR determines the necessary parameter alterations to react to a particular system state. [19] depicts an approach for CBR in DTs based on specific domain knowledge to automatically detect malfunctions in CPPSs. Case models, stored in the knowledge base, are used to identify a solution to compensate for undesired system behavior.

2.3. Model-driven Development

In model-driven development, the software is abstracted as models on which base source code is created automatically [20]. This enables non-software developers to design models and generate software. Applying model-driven development reduces interdisciplinary misunderstandings.

2.3.1. MontiArc

The modeling language for complex software architectures MontiArc [21] uses a small, easy-to-learn set of language features. MontiArc comprises: components, connectors, and configurations [22]. Components are computational units of the architecture model that leverage well-defined ports. Connectors establish communication between components by connecting these ports. This enables the modeling of software architectures with hierarchically structured and interconnected components. A code generator compiles MontiArc models into source code.

2.3.2. Self-adaptive Digital Twin Architecture

The quality-aware DT builds upon a reusable, self-adaptive DT reference architecture [19] in MontiArc, shown in Figure 1. The DT system consists of five DT components, an associated CPPS, a `Data Lake`, and the data flow. The system evaluation passes through the four main components in sequence. The output of a component is made available to the succeeding component as input. The components `Data Processor`, `Evaluator`, `Reasoner`, and `Executor`, realize the MAPE-K scheme, presented in Section 2.2. Figure 1 highlights the MAPE-K phases.

Workpiece creation generates data that is stored in the Data Lake. The `Data Processor` accesses this data and converts it into a DS. The `Evaluator` monitors the CPPS based on this DS. The `Reasoner` is notified if a malfunction is detected and provides a compensating improvement proposal based on knowledge about the CPPS and its desired system behavior. The `Executor` communicates the improvement proposal to the CPPS and supervises its execution. The CPPS provides feedback about the effectiveness of an improvement proposal. Thus, a steadily increasing `Knowledge Base` about a specific CPPS is established.

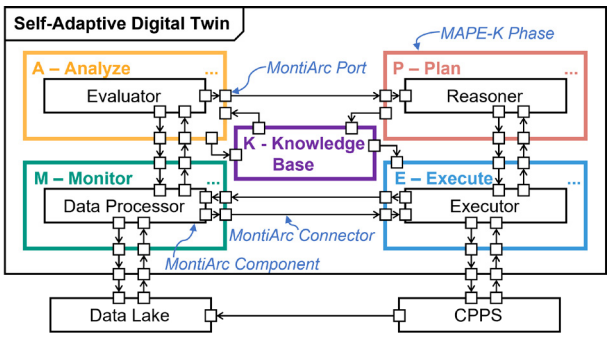

Figure 1. Self-adaptive Digital Twin reference architecture in MontiArc with MAPE-K phases, adapted from [19]

Multiple domains utilize DTs. Many DTs are individual implementations. To avoid reimplementation, [6] presents a model-driven development approach for a reusable DT reference architecture in MontiArc. This framework is deployed to create a DT for injection molding. The model-driven establishment of an OPC UA connection between a DT and the associated CPPS is depicted in [4]. A central information point with an integrated information model provides a bidirectional link between DT and CPPS.

2.4. Research Gap

This paper addresses the improvement of process quality based on a DT approach. Besides the application of DTs, other approaches to improving process quality are viable, e.g., using self-optimizing machining systems [23] or a version of predictive maintenance [24].

The model-driven DT for injection molding, presented in [6], does not consider the detection and improvement of process quality. It consists of a reusable DT reference architecture, which we modify to enable quality awareness. We integrate the concepts of ECG models [18] for the analysis of process data and CBR [19] for the planning of compensating measures in a self-adaptive DT framework based on MAPE-K. The model-driven OPC UA communication, depicted in [4], serves as a basis for the data exchange between DT and CPPS. All necessary technologies to develop a quality-aware DT exist. However, they are not yet used to improve process quality. In the following, we will utilize the available concepts to modify the existing DT reference architecture to enable quality awareness.

3. Enabling Quality Awareness in Digital Twins

This section presents a general process to develop quality-aware DTs for utilization in various use cases. It depicts the required steps to modify the existing self-adaptive DT reference architecture. Furthermore, alterations to enable OPC UA communication are described.

3.1. Architecture Modification for Quality Awareness

The self-adaptive DT reference architecture, presented in Section 2.3.2, does not support the observation of quality-relevant influences. The present state of the physical machine is unknown and not considered when assessing improvements. To mitigate this, a new `Quality Assessor` component is introduced to the reference architecture. Figure 2 shows the modified reference architecture in MontiArc. The `Quality Assessor` observes data produced during the manufacturing process and by the system environment. Common data points to observe are, e.g., motor current, axis position, and ambient temperature.

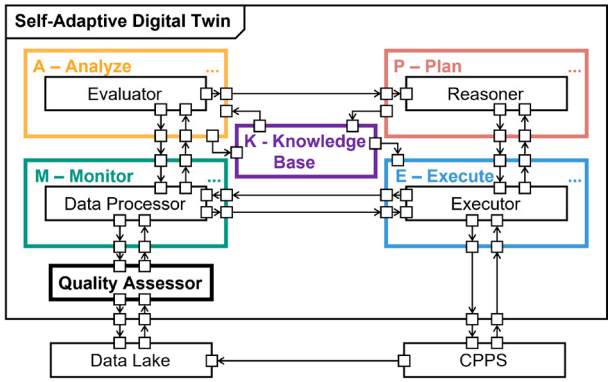

Figure 2. Self-adaptive Digital Twin reference architecture in MontiArc modified with the `Quality Assessor` component

During manufacturing, the CPPS and its surrounding environment produce data, that is collected, transmitted, and stored in a `Data Lake`. The `Data Processor` monitors the data by sending queries for specific data to the `Data Lake`. The `Data Lake` sends the requested data to the `Data Processor`. The `Data Processor` converts the data into a DS, which is readable and analyzable by the `Evaluator`. The `Evaluator` queries the DS to analyze the processed data, e.g., via an ECG model. If a process malfunction is detected, a correcting action is requested by sending an improvement goal compensating the error case to the `Reasoner`. The `Reasoner` determines a suitable approach to achieve the set improvement goal by drawing on previous experiences, which are available in the `Knowledge Base`. One option to identify a procedure for improving the machine settings is CBR. After determining an improvement case, the necessary parameter changes are sent to the `Executor`. The `Executor` then commands and monitors the execution of the parameter changes on the CPPS. The CPPS sends feedback about the success

or failure of a specific corrective action to the `Executor`. The `Executor` makes this feedback available to the `Data Processor`, `Reasoner`, and `Knowledge Base`. An extensive `Knowledge Base` is created through multiple cycles.

A unified modeling language (UML) activity diagram visualizes the loop through the modified DT reference architecture, shown in Figure 3. The activities are performed by one of the following four actors: the user, a human who manually interacts with the DT system; the CPPS, which manufactures a product; the `Quality Assessor`, the added component for quality assessment; and the DT, consisting of a MAPE-K loop. The activity diagram relies on ECG and CBR for analyzing and planning actions.

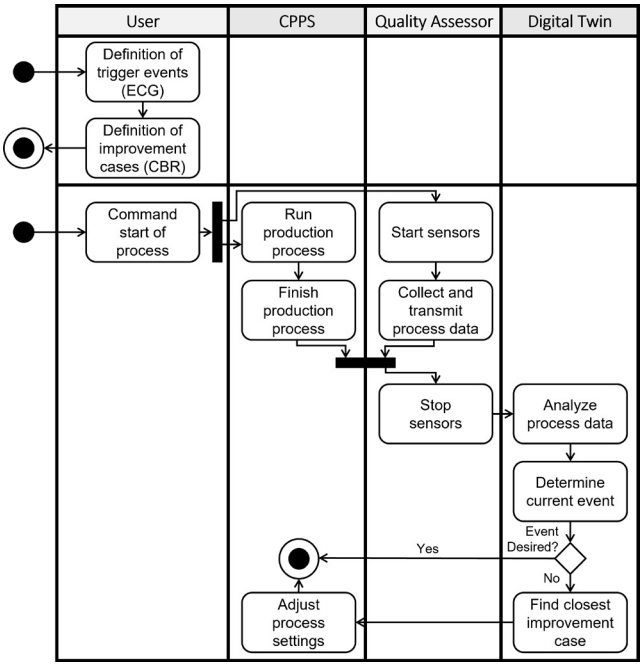

Figure 3. UML activity diagram of one loop through the modified Digital Twin reference architecture

Before a first loop through the DT, the user defines initial events for the ECG model and associated improvement cases used for CBR. The definition of additional events and cases is possible at a later stage. The manufacturing process can be started after the initial setup. This triggers the production of the product on the CPPS and activates the sensors of the `Quality Assessor`. The `Quality Assessor` sensors remain active to collect and transmit process data to the `Data Lake` until the CPPS completes the manufacturing process. The sensors of the `Quality Assessor` are turned off after the CPPS finishes production, and subsequently, the DT starts analyzing the available process data. The current quality-state is determined based on the collected data and through use of the ECG model. The detected state is either mapped to a desired or a non-desired event. When identifying a desired event, no adjustments to the machine settings are necessary, and the quality evaluation process terminates. At this point, the user can start a new manufacturing process.

If an undesired event is detected, an improvement goal is determined by the `Evaluator`. The `Reasoner` searches for a suitable improvement case to satisfy the stated goal. The CPPS's process settings are adjusted according to the improvement cases, and the quality evaluation is terminated. A new manufacturing process can start.

3.2. Incorporation of OPC UA Communication

The OPC UA communication link enables a decentralized and distributed application of the DT. Thus, communication between all components of the DT system independent of the system setup is possible. The integration of new sub-components into the `Data Processor` and `Executor` is required to incorporate OPC UA communication into the DT. Figure 4 shows the necessary modifications. Through the OPC UA link, the `Data Processor` receives process data and the `Executor` sends execution commands. To enable data input via OPC UA, a new sub-component called `OPCUADataProcessor` is implemented in the `Data Processor`. To ensure the data output via OPC UA, a new sub-component called `OPCUAExecutor` is implemented in the `Executor`. The task of the `OPCUADataProcessor` is to establish a reliable connection to the `Data Lake` and the data contained therein. The process data passes through the OPC UA link to the pre-existing `Data Processor` sub-component `DSCaster`, which converts the data into DSs. Subsequently, the DSs are processed by the following components without OPC UA related alterations. When a suggested improvement reaches the `Executor`, the existing `CBRExecutorLogic` prepares the execution command for process parameter changes. These changes are then communicated with the CPPS via the added `OPCUAExecutor` sub-component.

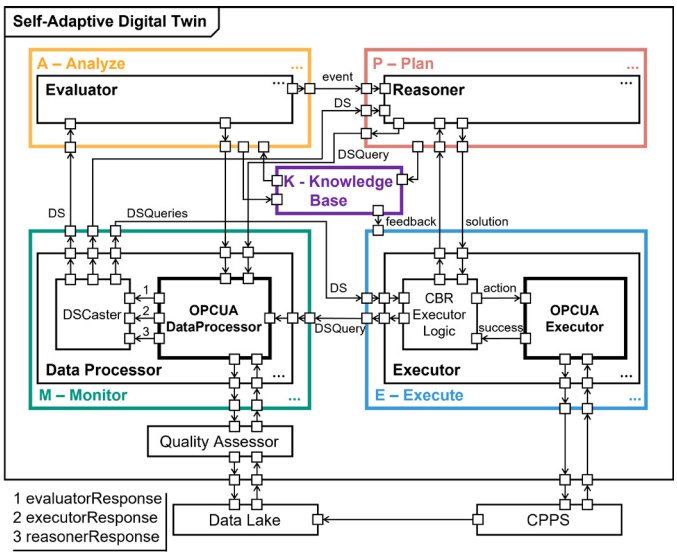

Figure 4. Self-adaptive Digital Twin reference architecture with integrated OPC UA communication sub-components

4. Application Example

As an application example, the modified DT reference architecture is realized for a laser system with OPC UA communication. For the programming of the DT, MontiArc, and Java 8 are employed. The OPC UA communication uses the no-code platform Collectu[1] and the open-source OPC UA framework Eclipse Milo[2]. The DT can operate on a physical TruPulse 2002 nano[3] fiber laser machine and on its virtual representation, simulated with the ISG-virtuos[4] simulation platform. The same Beckhoff TwinCAT[5] process control and DT are utilized for the physical and the virtual laser machine. Only one, either the physical or the virtual laser machine, can be connected to the process control; the shift occurs via a control button. This dual application enables the user to experiment with the virtual representation by testing scenarios that might potentially be dangerous to the user or damage the physical laser machine. Results found in tests on the virtual application can be deployed on the physical device. This enables a considerable increase in knowledge about the system and improves its testability and trainability.

Figure 5 shows the architecture of the application example. As a CPPS, the laser system is connected to the modified DT reference architecture. The laser system comprises the process control, the physical and the virtual laser machine. Data for quality assessment is collected via physical quality sensors to measure tangible process data or virtual quality sensors, which are built into the simulation of the virtual laser machine to collect virtually generated process data. Depending on whether the physical or virtual laser machine is connected, the associated quality sensors are deployed. Regardless of how data is collected, it is subsequently stored in the `Data Lake`. The four main components of the DT are adjusted to fit a use case of surveying and controlling the laser removal depth. The removal depth should lie within a predefined range. In case of deviations compensating measures are necessary. For this purpose, the `Evaluator` is equipped with a matching ECG model sub-component. A CBR sub-component of the `Reasoner` is adjusted to fit the use case.

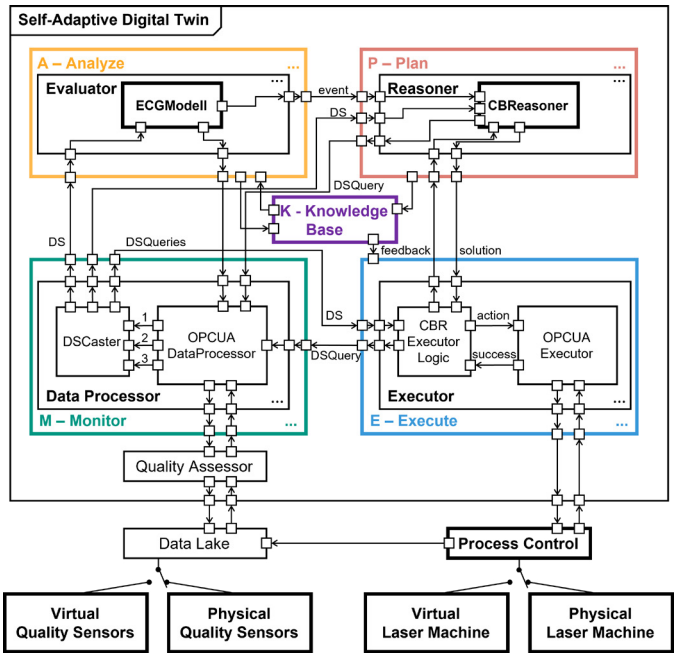

Figure 5. Self-adaptive Digital Twin reference architecture for connecting a physical and virtual laser machine

5. Discussion

The modifications to an existing self-adaptive DT reference architecture proposed in this paper result in a quality-aware DT. The modified DT reference architecture is suitable to reflect and compensate for changes in the performance and environment of the corresponding CPPS. Fundamentally, this was proven via an application example for a laser system. The realized quality-aware DT system establishes an OPC UA connection to the laser machine and the associated sensors for quality assessment. Based on the collected process data, the DT autonomously detects unintended system behavior within the limits of the use case and determines necessary system changes based on case similarity, which the process control subsequently executes. This application example only implements a rudimentary use case to demonstrate the fundamental suitability of the modified DT reference architecture. To obtain a comprehensive quality assessment of the CPPS, an expansion of the use case scope as well as vigorous testing and validation are necessary. Tangible refinement relevant for production purposes require the expansion and specification of Evaluator and Reasoner. Hereto, ECG and CBR can be supplemented or replaced by specialized methods, e.g., using machine learning approaches. Communication via OPC UA permits fast and low-effort connections to several different CPPSs. For analysis between two instances of a manufacturing process, OPC UA and the MAPE-K scheme are suitable. However, if improvements during production are necessary, the aptitude of OPC UA and MAPE-K is uncertain, since neither is designed to work in real-time. The shift between execution on a physical or virtual CPPS represents a genuine improvement in the use and build of a DT system's `Knowledge Base`. It is the basis for safe and improved testing and training of system functions. By exploiting the potential of both configurations, a fully tested DT system is created. Findings from experimental system improvements in virtual configuration can be used directly to operate the physical system.

[1] `https://collectu.de`
[2] `https://projects.eclipse.org/projects/iot.milo`
[3] `www.trumpf.com/products/laser/trupulse-nano/`
[4] `www.isg-stuttgart.de/en/products/isg-virtuos`
[5] `www.beckhoff.com/en-gb/products/automation/twincat/`

6. Summary and Outlook

This paper describes how an existing self-adaptive DT reference architecture is modified to reflect and compensate for changes in the performance and environment of a CPPS. This results in a quality-aware DT capable of reliably improving process quality. Instead of depending on idealized digital versions of a CPPS when analyzing and evaluating system behavior, the unique system conditions of a specific CPPS are taken into account. We modified the self-adaptive DT reference architecture, presented in [6], to include a component for collecting and assessing quality-relevant process data and enabling OPC UA communication between DT and CPPS. A use case demonstrates the capabilities of the quality-aware DT. The modified DT connects to a physical laser machine and its virtual simulation. Thus, enhancing system testability and trainability and quickly gaining comprehensive system knowledge.

The modified DT reference architecture for the improvement of process quality is not limited to the use of a specific CPPS or manufacturing technique. Milling operations, for example, are suitable for future applications. During ongoing research, the presented modified DT reference architecture is used to create a comprehensive application observing multiple quality-relevant factors. Part of this research will be determining conclusive indicators for process quality and tailoring error detection and resolution of these indicators in the `Evaluator` and `Reasoner` components, e.g., through machine learning methods. This enables adaptation to the unique CPPS characteristics.

Acknowledgement

The authors would like to thank the Ministry of Science, Research and Arts of the Federal State of Baden-Württemberg for the financial support of the projects within the InnovationsCampus Future Mobility (ICM).

References

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

[2] Boschert, S., Rosen, R., 2016. Digital Twin—The Simulation Aspect. Mechatronic Futures, 59–74.

[3] Kritzinger, W., Karner, M., Traar, G., Henjes, J., Sihn, W., 2018. Digital Twin in manufacturing: A categorical literature review and classification. IFAC-PapersOnLine, 51(11), 1016–1022.

[4] Kutscher, V., Olbort, J., Steinhauer, C., Anderl, R., 2020. Model-Based Interconnection of Digital and Physical Twins Using OPC UA. Advances in Manufacturing, Production Management and Process Control, 1216, 178–185.

[5] Cheng, B. H. C., de Lemos, R., Giese, H., Inverardi, P., Magee, J., 2009. Software Engineering for Self-Adaptive Systems: A Research Roadmap. Software Engineering for Self-Adaptive Systems, 1–26.

[6] Bibow, P., Dalibor, M., Hopmann, C., Mainz, B., Rumpe, B., Schmalzing, D., Schmitz, M., Wortmann, A., 2020. Model-Driven Development of a Digital Twin for Injection Molding. Advanced Information Systems Engineering, 12127, 85–100.

[7] Selic, B., 2003. The pragmatics of model-driven development. IEEE Software, 20(5), 19–25.

[8] Dalibor, M., Jansen, N., Rumpe, B., Schmalzing, D., Wachtmeister, L., Wimmer, M., Wortmann, A., 2022. A Cross-Domain Systematic Mapping Study on Software Engineering for Digital Twins. Journal of Systems and Software, 193, 111361.

[9] Stark, R., Thoben, K-D., Gerhard, D., Hick, H., Kirchner, E., Anderl, R., Wartzack, S., Krause, F.-L., Gräßler, I., Pottebaum, J., Schleich, B., Stelzer, R., Klein, P., Faustmann, C., Saske, B., Czwick, C., Gogineni, S., Klimmeck, L., Bajzek, M., Jacobs, G., Berroth, J., Zimmermann, T., Kranabitl, P., Göckel, N., 2020. WiGeP-Positionspapier - Digitaler Zwilling. ZWF Zeitschrift für Wirtschaftlichen Fabrikbetrieb, 115, 47–50.

[10] Preuveneers, D., Joosen, W., Ilie-Zudor, E., 2018. Robust Digital Twin Compositions for Industry 4.0 Smart Manufacturing Systems. IEEE 22nd International Enterprise Distributed Object Computing Workshop (EDOCW), 69–78.

[11] Eramo, R., Bordeleau, F., Combemale, B., Brand, M., Wimmer, M., Wortmann, A., 2022. Conceptualizing Digital Twins. IEEE Software, 39(2), 39–46.

[12] Lehner, D., Pfeiffer, J., Tinsel, E.-F., Strljic, Matthias M., Sint, S., Vierhauser, M., Wortmann, A., Wimmer, Ma., 2022. Digital Twin Platforms: Requirements, Capabilities, and Future Prospects. IEEE Software, 39(2), 53–61.

[13] Becker, F., Bibow, P., Dalibor, M., Gannouni, A., Hahn, V., Hopmann, C., Jarke, M., Koren, I., Kröger, M., Lipp, J., Maibaum, J., Michael, J., Rumpe, B., Sapel, P., Schäfer, N., Schmitz, G. J., Schuh, G., Wortmann, A., 2021. A Conceptual Model for Digital Shadows in Industry and Its Application. Conceptual Modeling, 271–281.

[14] Andersson, J., de Lemos, R., Malek S., Weyns, D., 2009. Modeling Dimensions of Self-Adaptive Software Systems. Software Engineering for Self-Adaptive Systems, 5525, 27–47.

[15] Sinreich, D., 2006. An architectural blueprint for autonomic computing. Autonomic Computing White Paper, 3.

[16] Arcaini, P., Riccobene, E., Scandurra, P., 2015. Modeling and Analyzing MAPE-K Feedback Loops for Self-Adaptation. IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, 13–23.

[17] Rutten, É., Marchand, N., Simon, D., 2017. Feedback Control as MAPE-K loop in Autonomic Computing. Software Engineering for Self-Adaptive Systems III. Assurances, 9640, 349–373.

[18] Almeida, E. E., Luntz, J. E., Tilbury, D. M., 2007. Event-Condition-Action Systems for Reconfigurable Logic Control. IEEE Transactions on Automation Science and Engineering, 4(2), 167–181.

[19] Bolender, T., Bürvenich, G., Dalibor, M., Rumpe, B., Wortmann, A., 2021. Self-Adaptive Manufacturing with Digital Twins. International Symposium on Software Engineering for Adaptive and Self-Managing Systems, 156-166.

[20] Mens, T., Van Gorp, P., 2006. A taxonomy of model transformation. Electronic notes in theoretical computer science, 152, 125–142.

[21] Medvidovic, N., Taylor, R. N., 2000. A classification and comparison framework for software architecture description languages. IEEE Transactions on software engineering, 26(1), 70–93.

[22] Butting, A., Haber, A., Hermerschmidt, L., Kautz, O., Rumpe, B., Wortmann, A., 2017. Systematic Language Extension Mechanisms for the MontiArc Architecture Description Language. European Conference on Modelling Foundations and Applications, 10376, 53–70.

[23] Möhring, H.-C., Wiederkehr, P., Erkorkmaz, K., Kakinuma, Y., 2020. Self-optimizing machining systems. CIRP Annals, 69(2), 740–763.

[24] Ran, Y., Zhou, X., Lin, P., Wen, Y., Deng, R., 2019. A Survey of Predictive Maintenance: Systems, Purposes and Approaches.