Key Engineering Materials
ISSN: 1662-9795, Vol. 1049, pp 43-54
doi:10.4028/p-YFY9Oq

Submitted: 2026-01-16
Revised: 2026-02-24
Accepted: 2026-03-02
Online: 2026-04-15

Method of an Automated Tool Design of Punch-Bending Processes Using the Asset Administration Shell

Henning Peters[1,a*], Andreas Mazur[2,b], Brijesh Chaudhary[1,c], Lukas Kersting[1,d], Ansgar Trächtler[1,3,e], Barbara Hammer[4,f]

[1]Fraunhofer Institute for Mechatronic Systems Design (IEM), Paderborn, Germany

[2]Bielefeld University, Center for Cognitive Interaction Technology (CITEC), Germany

[3]Heinz Nixdorf Institute (HNI), Paderborn University, Germany

[a]henning.peters@iem.fraunhofer.de, [b]amazur@techfak.uni-bielefeld.de, [c]brijesh.chaudhary@iem-extern.fraunhofer.de, [d]lukas.kersting@iem.fraunhofer.de, [e]ansgar.traechtler@hni.uni-paderborn.de, [f]bhammer@techfak.uni-bielefeld.de

*Corresponding author

Keywords: punch-bending, digital twin, asset administration shell, process optimization.

Abstract. A common challenge of punch-bending is the control and compensation of process deviations which are induced, e.g., by different material batches. The resulting deviations are currently countered iteratively using expert knowledge in the design phase as well as during the production process. However, this is not always successful due to the complex interactions between semi-finished product properties and forming tools. To automate the optimization of tool geometries and process adjustments, an accurate prediction model of correlations between process/tool parameters and product properties is necessary. In that regard, an application programming interface (API) aligned with the Asset Administration Shell (AAS) specification is developed utilizing a hybrid data-driven approach. It enables the transformation of various data formats e.g., from sensor and simulation data into a machine-readable structure. The API is linked to a digital twin and a database, to automatically gather requested information and provide the new structured data to a machine learning (ML) model. To validate the developed method, hybrid punch-bending data is automatically transferred to an ML model which then returns tool geometry suggestions for a defined product geometry.

Introduction

Punch-bending is a forming process in which semi-finished products like metal strips or wires are formed into products like brackets, mounts, electronic contacts or spring elements. The respective punch-bending machines consist of several bending operations to achieve complex geometries [1, 2].

A common challenge of punch-bending is the control and compensation of process deviations. Those are induced in the wire and therefore in the process in form of curvature and residual stresses during wire production and wire coiling [1, 3]. In industrial applications, those deviations are currently countered iteratively using expert knowledge and setup guidelines in the design phase as well as during the production process. However, this is not always successful due to the complex interactions between semi-finished product properties and forming tools [1, 4, 5].

To enable the optimization of tool geometries and process adjustments, an accurate prediction model of correlations between process/tool parameters and product properties is necessary. Therefore, a hybrid data-driven modeling approach including classical models (analytical, empirical, numerical) and real data from the manufacturing process is chosen. Classical models enable investigations using a wide range of tool configurations without having to manufacture them, as well as capturing effects that are difficult or impossible to measure. However, they are based on assumptions and simplifications, which means that deviations from reality may occur [3, 6]. By including real process data within the data-driven approach, deviations and disturbances which are not considered by classical modeling approaches can extend the data set to receive a more holistic understanding of the

process and therefore a more robust model [7]. However, this hybrid approach requires a large amount of data in different data formats. In this regard, an automated acquisition, preprocessing and transfer of hybrid data is required, which can be realized by a digital twin (DT).

The first step towards automated data acquisition was presented by the authors in [8], by defining a possible structure of the DT as well as by connecting the real process with the DT. However, this current approach only includes real process data. To add further data sources regarding the hybrid approach, an automated integration of hybrid data into the DT has to be investigated.

Related Work

The concept of the digital twin is often seen as an enabler for IIoT and Industry 4.0 applications. However, because there are several different definitions regarding the concept of the digital twin, only a very specific and limited utilization is possible, which makes it significantly more difficult to transfer to other applications. To counteract this, multiple standardized specifications were developed to enable an interoperability between different assets. Via semantic modeling the specifications describe the properties and the behavior of an asset as well as the relationship to other assets [9, 10]. Depending on the specification they also provide defined interfaces to interact with the digital representation [11].

The Asset Administration Shell (AAS) is one of the developed specifications which is subsequently refined by the Industrial Digital Twin Association (IDTA) [10]. In [12] the combination of multimodal, i.e. hybrid, data and its integration into automation structures is defined as research challenge for data-driven modeling. Due to the defined technology-neutral standard of the AAS, interoperability with regard to the exchange of information and data from different sources is made possible [13]. Therefore, the AAS can be used to automatically acquire hybrid data from the punch-bending process and thus use it for the hybrid data-driven modeling approach.

Several cases have already been examined with regard to the utilization of the AAS in the context of Industry 4.0 applications. One focus of research is manufacturer- and supplier-independent data exchange. Here, the technology-neutral nature of AAS is utilized to enable shared production and development between multiple stakeholders [14, 15]. In this regard, the concept of the digital product passport represents an implementation of this data exchange. [16]. The aspect of manufacturer- and supplier-independent data exchange between different machines in production is another research topic, which enables the interoperability between automated industrial systems [17, 18]. Other prominent applications include skill-based manufacturing [19], process planning [20], process monitoring [21] and predictive maintenance [22].

However, to the best of the authors knowledge, the application regarding a hybrid data-driven tool design in forming technology has not yet been researched.

Data Sources

In order to pursue the hybrid data-driven approach to optimizing tool design for the punch-bending process, a large amount of data in various formats is required. The hybrid data used in this paper can be divided into three groups, which are shown in Table 1: real process data, simulation data and historical data. The table also shows the quantities contained in each data source as well as their data formats. The table also lists the tool configurations that are considered as input data and are available in the same data format for each previous data source.

Table 1. Data sources, quantities and data formats of the hyrid data-driven modeling approach of punch-bending processes.

Data Source	Quantity	Data Format
Process data (sensor data)	Forces, bending angle, positions, curvature	Time series data, single sensor data, image data
Simulation	Forces, bending angle, residual stresses	Time series data, single sensor data, nodal data
Historical data (analytical model, reference studies)	Forces, bending angle	Tabular data
Tool configuration	Tool geometries, process settings	Single parameter values

Process data is generated within the real punch-bending process which is displayed in Figure 1. It consists of a material coil, a mechatronic straightener and a punch-bending machine. The mechatronic straightener includes seven straightening rolls. The three upper rolls are each connected to a stepper motor and can therefore be moved via the machine control. Within this punch-bending process, multiple sensors are installed to capture the effects appearing during the straightening and bending operations. The straightening machine is equipped with laser triangulation sensors to determine the positions of the upper straightening rolls. Load cells are also attached to the upper rolls to measure the resulting straightening forces. Both the position and force signals are sent to the machine control as time series data. This is implemented using Beckhoff's TwinCAT automation software. Using an optical measuring system, the wire curvature can be determined offline. First, several coordinate points on the wire geometry are determined and then converted into a corresponding curvature. This is then available as a single sensor data point. The bending angle of the punch-bending product is determined inline using an optical measuring system as well. The image recognition of the camera software identifies the relevant geometric features, outputs them as single sensor data and forwards them afterwards to the automation software.

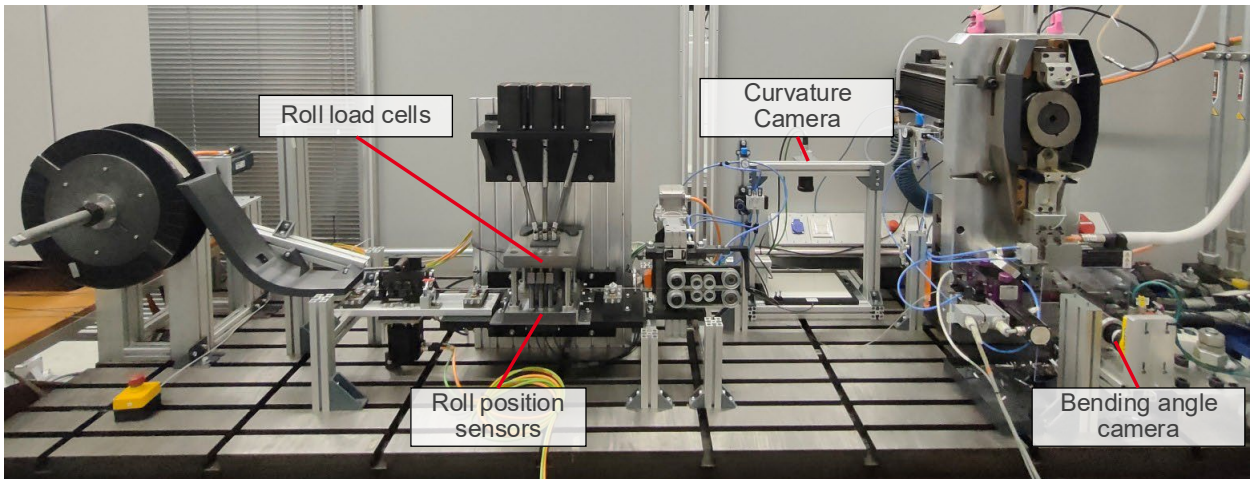

Fig. 1. Overview of the punch-bending process, which is used for process data generation.

Simulation data is generated via the FE-Software Abaqus. Besides forces and the bending angle also residual stress or strain is captured. Here, the calculated values are assigned to the individual nodes of the mesh so that the determined effects are available as nodal data. For historical data, data from analytical models and data from experimental reference studies are summarized. In this way, previously gathered knowledge can also be incorporated into data-driven modelling. It is assumed that this data is available in tabular form.

The hybrid data-driven approach for optimizing the tool design for punch-bending processes thus has several different data formats, which, given the large amount of data required, necessitates automated acquisition, preprocessing and transfer of the hybrid data in a uniform machine-readable

format. This could be realized by a digital twin and a suitable automation structure. In [8], the authors already described a structure between the digital twin, its included database and the real punch-bending process. Consequently, only real data was included. To enable the processing of hybrid data, an extension of the existing automation approach is necessary.

Modification of the Automation Structure for Hybrid Data

The modification approach is shown in Figure 2. Regarding this hybrid data-driven approach, simulation data as well as historical data are to be integrated in the same machine-readable format to enable the processing by subsequent clients like preprocessing or the ML module.

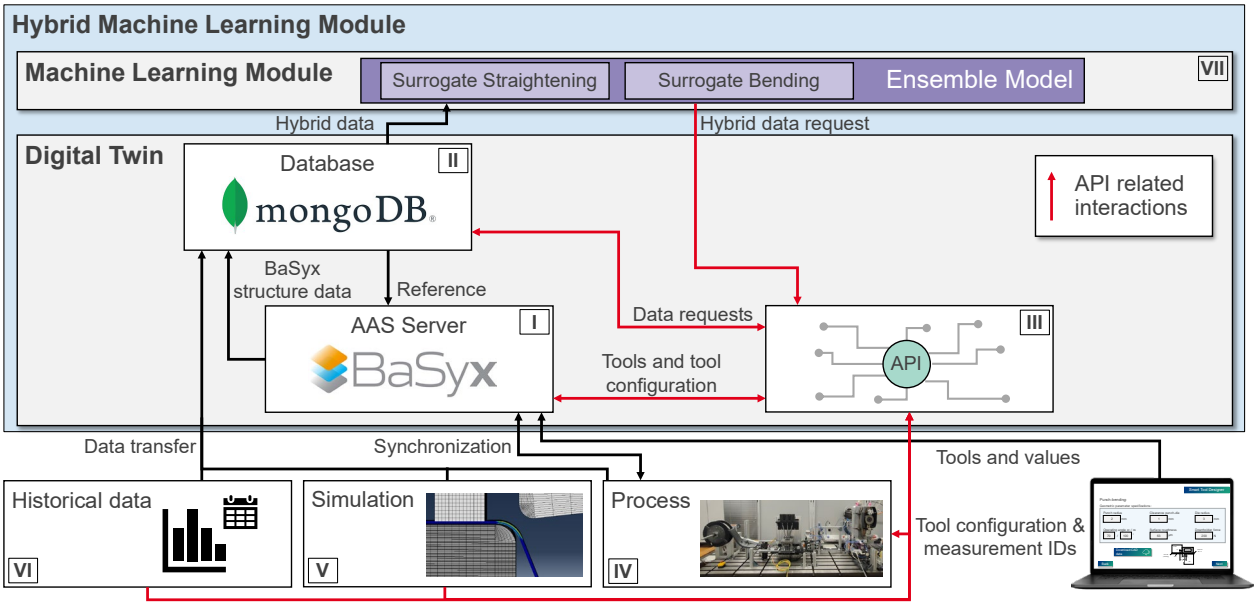

Fig. 2. Schematic representation of the hybrid automation structure, incorporating the API to enable the transformation of various data formats into a machine-readable structure.

As described in previous work the AAS specification is integrated by using the software called AASX Package Explorer. It enables the development of the digital representation of the punch-bending process and its assets. Furthermore, additional meta data of the assets can be stored in the respective submodels to obtain a holistic representation and comprehensive information storage. This information model is then uploaded to the AAS Server (**I**). There are several open-source AAS implementations. An overview and comparison is given in [23]. Based on the study conducted in [23], it is evident that Eclipse BaSyx matches the requirements for usage within the hybrid data-driven model best among the implementations examined. Therefore, it will be used as the implementation of the AAS for the digital twin in this paper which allows the interaction with the created digital representation. For the digital twin of the punch-bending process the communication was implemented via HTTP/REST in a Docker environment. Here, two versions were developed: A local/unsecured and a secured version. All components therefore can be either addressed by their local port or via a nginx reverse proxy to also implement secured communication via HTTPS.

An important subcomponent of the digital twin is the database (**II**). It has to store all acquired process data as well as initial process and tool parameters which are then transferred to the ML module. BaSyx also requires a database to store the AAS meta data. For this, the NoSQL data management system MongoDB is currently the only persistent data storage implementation in BaSyx (V2). Due to the properties of a NoSQL database, MongoDB enables the storage of multiple data formats. Also, time series data can be stored in so called time series collections, which offer an optimized storage for time series data compared to the standard MongoDB collections. Therefore, MongoDB was also selected for the storage of process data, process parameters and meta data of the punch-bending process. However, in order to combine the various data formats from different data sources into a standardized format for hybrid data-driven modelling, a database structure was

developed to assign the used tool configuration to the respective experiment or simulation. This is realized by an identification system which is shown in Figure 3. First, each tool or process property serving as input data receives its own Input ID (marked in yellow). Tool and process properties include, e.g., straightening roll infeed, the radius of the die or the angle of the bending punch, but also material properties. Sensor signals, simulation results or results from reference studies represent output data and are assigned a Output ID accordingly. The IDs with their corresponding assignments are then stored in a respective database collection. The Input Parameter ID combines a Input ID with a specific value (marked in turquoise). The combination of all tool parameters of one measurement is then summarized in a Input Configuration ID (marked in brown). If one or multiple tool parameters are changing, new Input Parameter IDs as well as a new Input Configuration ID, if not already existing, are created. With the initialization of a new measurement, a new Measurement ID is created and linked to a specific Input Configuration ID to ensure that recorded sensor data is linked to the tool and process properties (marked in pruple). Acquired data of the sensors, either real or simulated, are then assigned to the Measurement ID and their respective Output ID within the so-called Output Data Collection (marked in light green).

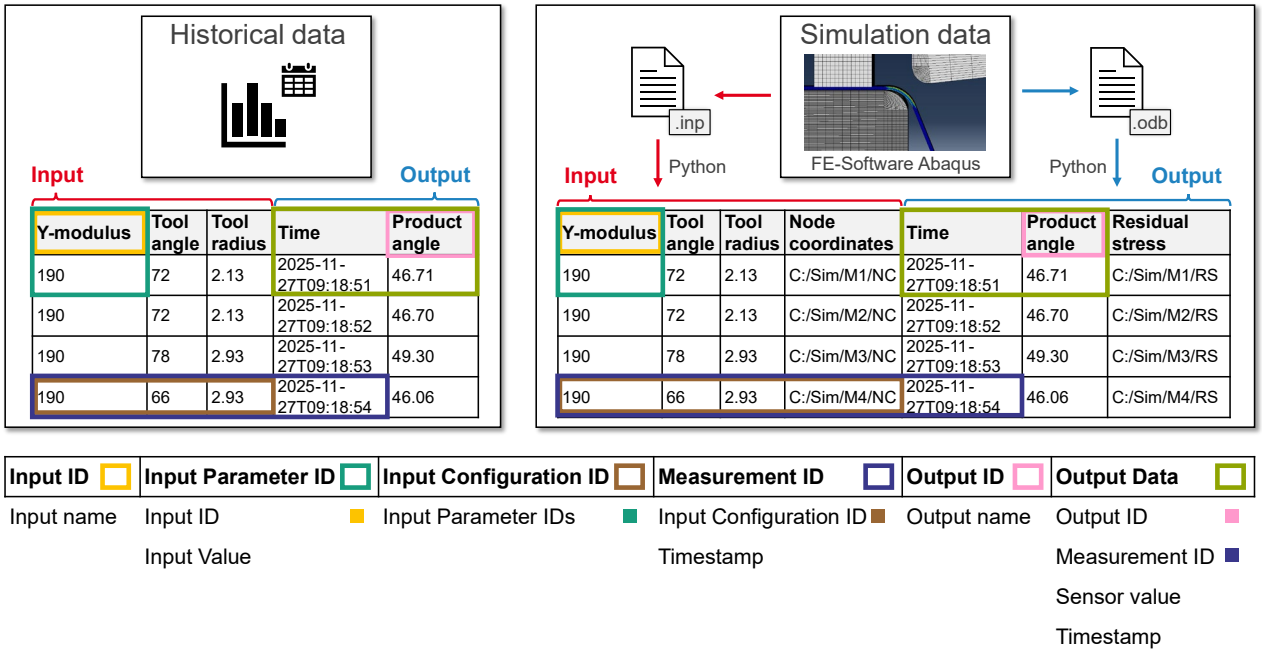

Fig. 3. Schematic representation of the identification system which enables a standardized format for hybrid data-driven modeling by transforming parameter and values into Input, Input Parameter, Input Configuration, Output and Measurement IDs.

As described in [8] the tool and process properties are listed in the digital twin, where they are extracted. Before an experiment is conducted on the real system, the values for each tool are entered by the operator, so that the IDs can be generated. This was conducted via the TwinCAT automation software of the machine. For the mere consideration of real sensor data this approach was sufficient. However, when considering the inclusion of hybrid data sources, which means the addition of simulation and historical data, this procedure is no longer reasonable. This is because the processing of simulated and historical data would otherwise depend on the machine control, even if it is not needed at that point in time. Furthermore, the generation of IDs via the machine control was subject to a certain degree of failure due to the cyclical processing of the automation software. Therefore, a centralized and modular approach is necessary which is displayed schematically with the application programming interface (API) (**III**) in Figure 2. The API is created in Python, ensuring the transferability of the methodology. The primary function of the API is to adopt the mechanisms which were implemented in the automation software by generating and assigning IDs for real process data as well as for simulation and historical data. The API centralizes the transformation of hybrid data formats into a standardized, machine-readable format, thereby enabling hybrid data-driven modeling

of punch-bending processes. Therefore, a connection between API and AAS server is set up. Furthermore, the API also acts as an interface to the database. For example, requests regarding specific IDs can be addressed.

Process data (**IV**) is recorded and transferred via the TwinCAT automation software. Within the automation software, a connection to the database is set up using the Database Server function. For each sensor, so-called AutoLog groups are created which enable an automated transfer of the process data to the database. The IDs provided by the API are also included in the data logging of the respective sensor signals so that the data can be assigned to the respective measurement. Simulation data (**V**) of the punch-bending process is created with the FE-software Abaqus. As shown schematically in Figure 3, the simulation data for each simulation is stored in an inp and an odb-file. The inp-file is a text document containing the input data for the simulation. The odb-file, on the other hand, is a proprietary file containing the simulation results. Here, the data has to be transformed into tabular data, which is then inserted into the database structure and the AAS server via the API. Historical data (**VI**) is already provided in tabular form. Accordingly, this data is transferred to the database structure and the AAS server using the same systematic approach via a Python script and the API.

Subsequently, the acquired hybrid data is transferred to the ML module (**VII**) which contains an ML ensemble model as well as an algorithm to compute counterfactual explanations. The ML algorithms allow the generation of process- and tool design suggestions, as already presented by the authors in [24]. Suggested process parameter adjustments are then returned to the digital twin to implement direct adjustments like straightening roller infeed to the system, or to forward tool geometries to the operator. The following section will go into more detail about the functionality of the automated data-driven modeling of punch-bending processes, from initializing a measurement to receiving a setup suggestion.

Method of Automated Data Processing for Hybrid Data-Driven Models

To achieve an automated data processing of hybrid data the approach is divided into a procedure for process data and a procedure for simulation and historical data. This is because the real process data is transferred to the database and the digital twin via the automation software, while simulation and historical data is transferred via data tables.

Procedure for process data.

Within this procedure, seen in Figure 4, the section Recording sensor data is based on the methodology previously implemented in the automation software, which was described by the authors in [8]. Instead of the automation software, the API is utilized to generate the respective IDs. First, measurements are initiated via the digital twin by specifying the input parameters which are to be examined. Using the synchronization function between the digital twin and the system the parameters are transferred to the system which requests the Input, Input Parameter, Input Configuration, Output and Measurement ID using the respective HTTP requests defined by the API. If an ID is non-existent, a new ID is generated and inserted into the designated database collection. Otherwise, the existing ID is returned. Additionally, the initialized measurement as well as its meta data like the data source is listed as a segment in the digital twin to be accessible by the ML module.

After one or multiple measurements are completed, in section Transfer data to ML module, the ML module checks for new measurements in the digital twin. For that the ML module requests the API to initiate the search for new Measurement IDs which are forwarded to the digital twin. Using an additional database collection, it is checked whether a measurement was already sent to the ML module or not. If this is not the case, the new Measurement IDs are sent back to the API. Afterwards, the data encrypted using IDs is translated into its plain names and actual values before being transmitted as a list of dictionaries to the ML module.

Depending on whether the ML module has been initialized, two possible scenarios must be considered: scenario (a) describes the case in which the ML module has never been used before. Here, all gathered data is used to train the ensemble model. Scenario (b) represents the case in which the

ML module has already been initialized. Then, only new data that so far has not been used to train the model is used for finetuning. Using the trained ensemble model, the algorithm for computing counterfactual explanations can search for a possible process configuration suggestion, given a desired target geometry. Afterwards, the suggestion is sent back to the digital twin.

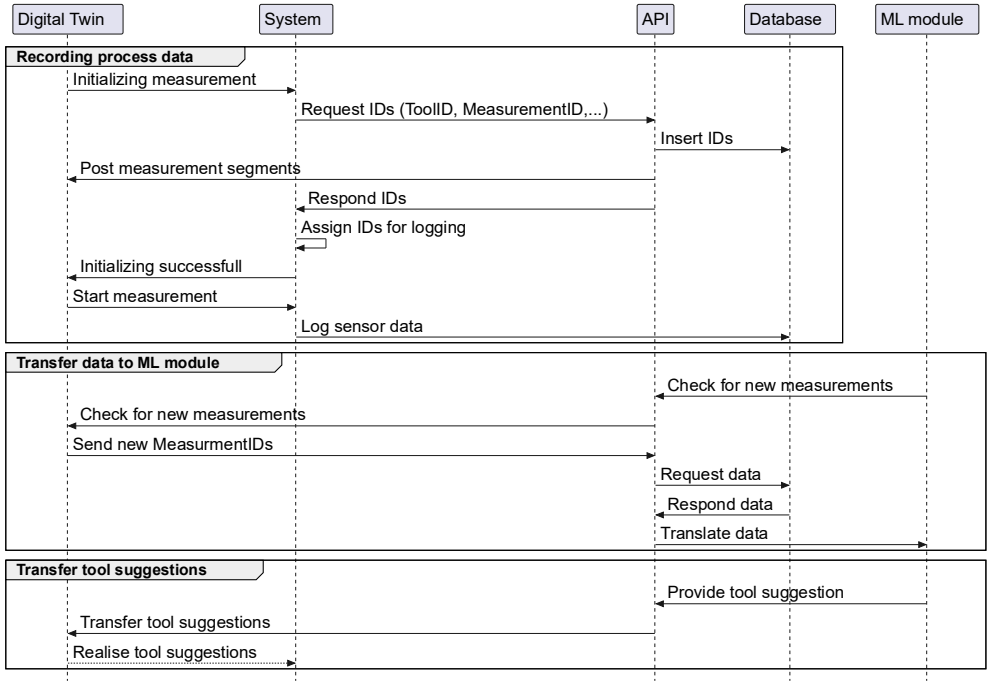

Fig. 4. Sequence diagram of the developed method of automated data processing for hybrid data-driven models using process data.

Procedure for historical and simulation data.

For historical and simulation data the first section differs compared to the procedure for process data, which is displayed in Figure 5. Where process data is transferred directly to the database and sorted into the database structure, historical and simulation data must first be imported and then converted into the database structure. Due to the different data formats, the data has to be provided in a uniform tabular format like a csv-file, which is the case for the historical data used. The simulation data of the punch-bending process, on the other hand, is acquired with the FEM-Software Abaqus. The simulation data for each simulation, as mentioned before, is stored in an inp and an odb-file. The inp-file contains input data like material properties, tool geometries or process settings in a text document format. The proprietary odb-file, contains the simulation output like the product angle, forming forces or residual stresses. Both files are first converted by a Python script, using the Python library abqpy [25] for the odb-file. Here, the data is transformed into tabular data, which is then inserted into the database structure and the AAS server via the API. The data table, either historical or simulation data, then provides the necessary data to the API. Each line represents a measurement taken (cf. Figure 3). Therefore, the input has to be separated from the output data so that the ML module is able to learn the resulting correlations. For this purpose, keywords were defined for input and output data, which are compared with the keys used for the historical or simulation data. After that, input and output parameters are assigned to the Input Collection and respectively the Output Collection with their specific IDs, which are generated by the API. Accordingly, the Input Parameter and Input Configuration IDs are generated. For each line, one Measurement ID is assigned. Similar to process data, the Measurement IDs and the respective meta data are posted into the digital twin, so that the ML module can access the measurements. Subsequently, the historical or simulation data is stored in the Sensor Data Collection.

The sections Transfer data to ML module and Transfer tool suggestions are almost identical compared to the procedure for process data (cf. Figure 4). The Measurement IDs are requested by the

ML module. They are then translated by the API into the actual keys and values, so that the ML module, after learning the correlations, can provide tool suggestions to be adopted by the operator.

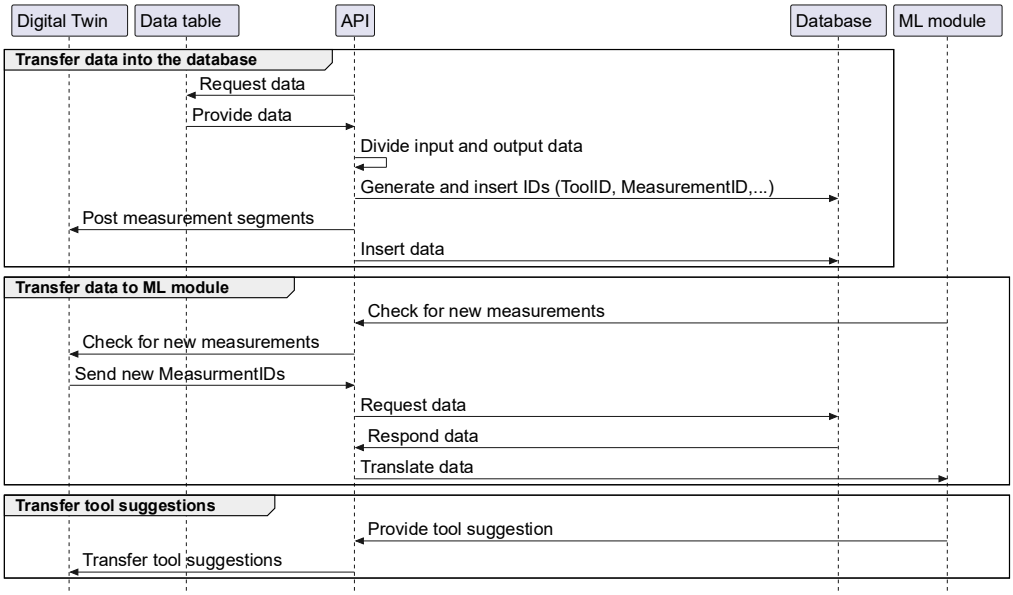

Fig. 5. Sequence diagram of the developed method of automated data processing for hybrid data-driven models using simulation and historical data.

Validation of the Method

To validate the method described, a case study was conducted using the developed ML model and data set presented by the authors in [24]. The data set falls into the category of historical data and displays a single-stage bending operation of a punch-bending process. The input parameters used are the Young's modulus, the wire thickness as well as the angle and radius of the bending tools. As outcome serves the resulting angle of the punch-bending product. Because it is a historical data set, it already comes in a tabular format (csv) (see data sources). Therefore, it does not need to be transformed as in the case of simulation data.

Validation procedure.

Following the developed procedure (cf. Figure 5), the data is inserted via a Python script. Depending on the file name and the parameters used, the data set is sorted into either the Simulation or Historical data category based on predefined keywords. These keywords include data format-specific terms that can appear both in the file name and among the listed input or output parameters. Because the data set, e.g., lacks keywords regarding node coordinates as output parameters, the data set considered in this paper is correctly assigned to the category historical. Compared to the retrospective assignment of the data source for historical and simulation data, process data is assigned to the Real Process category at the moment a measurement is initialized. This is achieved through the continuous connection of the API to the real process and its data acquisition. This meta data can later be used by the ML module to differentiate between the data of the hybrid data sources. In a similar way the parameters are compared with input-specific or output-specific keywords in order to categorize the parameters as input or output data. Accordingly, the input parameters Young's modulus, wire thickness, tool angle and tool radius are assigned to the Input Collection and the output parameter product angle to the Output Collection. The respective IDs as well as the Input Parameter, Input Configuration, Measurement and Output IDs are then generated by the API.

To represent the imported measurement within the digital twin, the IDTA time series data submodel template [26] is used. For this work the focus lies on the own added submodel element Source which displays the data source of the data set, which is in this case historical (cf. Figure 6, step I). Each measurement is therefore posted into the digital twin as submodel element collection

containing the respective submodel elements. Afterwards, the data is inserted into the database structure.

After this procedure, the data sets of every data source are stored in the same data format within the developed database structure. The subsequent steps are therefore identical for all data sources and can thus be applied to the validation of the methodology for automated processing of hybrid data. Within the scope of this paper, the focus will therefore remain on the historical data set from [24]. Accordingly, the measurement list within the digital twin is checked for recently added Measurement IDs. Using an additional database collection, the measurement list of the digital twin is compared with already transferred Measurement IDs to prevent duplicating data sets. New Measurement IDs are then sent to the API, which requests the respective incorporated IDs to translate those to actual names and values, before they are transferred to the ML module. The data is used to train a surrogate model enabling the calculation of counterfactual explanations (cf. Figure 6, step II). Using the trained model, a target angle is inserted (Figure 6, step III) to generate process configuration suggestions based on the counterfactual explanations (cf. Figure 6, step IV). A target product angle of 120° leads, e.g., to a process configuration displayed in Figure 6 (step V) which is sent to BaSyx. This completes the developed procedure. Thus, a data set has been successfully integrated into the database structure and then automatically transferred to the ML module in order to eventually obtain a prediction for a suitable process configuration for the punch-bending process.

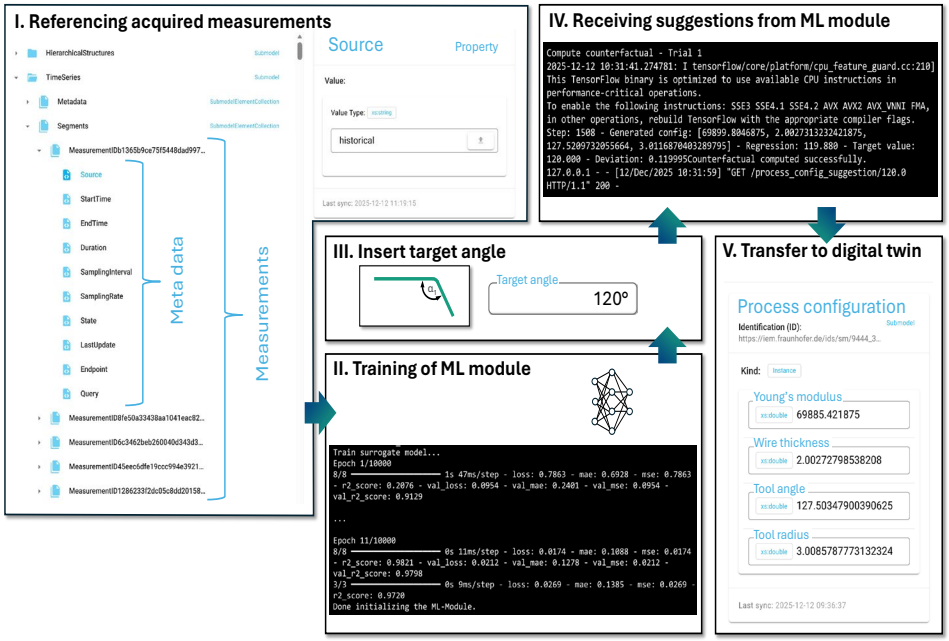

Fig. 6. Validation process for automated data processing for hybrid data-driven modelling of punch-bending processes: From automated integration via the ML module to the final parameter suggestion in the digital twin.

Discussion

The results successfully show the validity of the presented automation approach for punch bending. The import and transfer of data set used for validation, containing 5 parameters and 663 data points in the context of the experiment, takes around 34 seconds. As an example, a simulation data set of a single-staged bending operation provided with Abaqus, containing 26 parameters and 663 data points takes around 196 seconds. When repeating the step while retaining the previous data, only a small increase in time of 1–2 seconds is recorded for the historical data, while the simulation data already requires 260 seconds for the repetition. One limiting factor of this approach is the current reference of measurements in the digital twin. BaSyx stores such a submodel and all its contents in a single MongoDB document in BSON format, which has a maximum capacity of 16 MB. Accordingly, this approach reaches its limits with larger data sets.

Conclusion and Outlook

The article presented a method of automated data processing for optimizing the tool design of punch-bending processes by using a hybrid data-driven approach. Previous work initially introduced the digital twin based on the AAS specification as an automation concept. In order to complement the hybrid data-driven approach, the previously developed integration of process data into the automation structure was supplemented by simulation and historical data. Therefore, an API was developed which acts as an interface between hybrid data sources, the database and the AAS server. The API centralizes the transformation of hybrid data formats into a standardized, machine-readable format, thereby enabling a hybrid data-driven modeling of punch-bending processes to achieve an automated tool design. Furthermore, a direct connection between ML module to the digital twin has been established. It was demonstrated using the developed methodology, that due to the standardized format process parameters and product properties of the punch-bending process can be automatically transmitted to the ML module. The correlations learned in this way then resulted in a suggested process configuration for a specified target angle. Automatically returned to the digital twin, the suggested process configuration can then be adopted by the operator to optimize, e.g., the geometry of the punch-bending tool. The developed method of automated data processing thus enables the implementation of an automated process and tool design for punch-bending processes. Thanks to its modular structure, a transfer to other forming processes is also conceivable.

However, the method presented in this paper still has some limitations. For example, the approach is limited to tabular data. Accordingly, each data record must be provided in a tabular format or converted into such a format, as presented for simulation data. Such a conversion can take some time to complete. In the case of converting simulation data, e.g., the version of the odb-files must first be checked and updated if necessary. These topics will be further investigated in future work. Another aspect of future work is the representation of time series data in simulation data. Because every simulation is represented in a single line of the table, time series data for now are displayed by a reference to a specific path in a local folder structure. It is intended to adapt to the structure of the process data by listing individual time steps as well for the simulation data. The referencing of measurements within the digital twin in general must be adjusted as well, to enable the integration of large amounts of data. This is limited by BaSyx in terms of storage capacity by storing the contents of the time series submodel within a single MongoDB document.

Acknowledgment

The authors gratefully acknowledge the funding by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) for the project “Data-Driven Modelling of Metal Bending Processes” (project no. 520459685), funded in the frame of the DFG priority program 2422 “Data-driven process modelling in metal forming technology" (500936349).

References

[1] M. Paech, Advanced semi-automatic straightening technology, Wire Journal International (2008) 74–79.

[2] K. Richter, F. Reuther, R. Müller, D. Landgrebe, Investigating the Influence of Bending Parameters on the Springback Behavior of Ultra-High Strength Spring Strips, MSF (2018) 125–133. https://doi.org/10.4028/www.scientific.net/MSF.918.125.

[3] M. Grüber, Konzepte zur Steuerung des Richtwalzprozesses bei variierenden Richtguteigenschaften, PhD thesis, Rheinisch-Westfälische Technische Hochschule Aachen, Aachen, 2019. ISBN: 978-3-95886-307-1.

[4] A. Amor, M. Rachik, H. Sfar, Combination of Finite-Element and Semi-Analytical Models for Sheet Metal Leveling Simulation, KEM (2011) 182–189. https://doi.org/10.4028/www.scientific.net/KEM.473.182.

[5] M. Gräler, R. Springer, C. Henke, A. Trächtler, W. Homberg, Assisted setup of forming processes: compensation of initial stochastic disturbances, Procedia Manufacturing 25 (2018) 358–364. https://doi.org/10.1016/j.promfg.2018.06.104.

[6] L. Steinweder, A.J. Kainz, K. Krimpelstaetter, K. Zeman, Numerical Simulation of Tension Losses and Reaction Forces in Tension Levellers, Steel Research Intl., Special Edition: 10th ICTP (2011) 343–348.

[7] M. Liewald, T. Bergs, P. Groche, B.-A. Behrens, D. Briesenick, M. Müller, P. Niemietz, C. Kubik, F. Müller, Perspectives on data-driven models and its potentials in metal forming and blanking technologies, Production Engineering Research and Development 16 (2022) 607–625. https://doi.org/10.1007/s11740-022-01115-0.

[8] H. Peters, A. Mazur, A. Trächtler, B. Hammer, Integration of a digital twin for data-driven modeling of punch-bending processes using the asset administration shell, in: P. Carlone, L. Filice, D. Umbrello (Eds.), Material Forming: ESAFORM 2025, Materials Research Forum, 2025, pp. 1538–1547. https://doi.org/10.21741/9781644903599-166.

[9] M. Jacoby, T. Usländer, Digital Twin and Internet of Things - Current Standards Landscape, Applied Sciences 10 (2020) 6519. https://doi.org/10.3390/app10186519.

[10] B. Boss, S. Malakuti, S.-W. Lin, T. Usländer, E. Clauer, M. Hoffmeister, L. Stojanovic, Digital Twin and Asset Administration Shell Conepts and Application in the Industrial Internet and Industrie 4.0: An Industrial Internet Consortium and Plattform Industrie 4.0 Whitepaper (2020). https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/Publikation/Digital-Twin-and-Asset-Administration-Shell-Concepts.pdf.

[11] T. Miny, M. Thies, L. Lukic, S. Käbisch, K. Oladipupo, C. Diedrich, T. Kleinert, Overview and Comparison of Asset Information Model Standards, IEEE Access 11 (2023) 99189–99221. https://doi.org/10.1109/ACCESS.2023.3312286.

[12] S. Kamm, S.S. Veekati, T. Müller, N. Jazdi, M. Weyrich, A survey on machine learning based analysis of heterogeneous data in industrial automation, Computers in Industry 149 (2023). https://doi.org/10.1016/j.compind.2023.103930.

[13] The Structure of the Administration Shell: Trilateral Perspectives from France, Italy and Germany (2018). https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/Publikation/hm-2018-trilaterale-coop.html.

[14] M. Volkmann, A. Wagner, J. Hermann, M. Ruskowski, Asset Administration Shells and GAIA-X Enabled Shared Production Scenario, in: F.J.G. Silva, L.P. Ferreira, J.C. Sá, M.T. Pereira, C.M.A. Pinto (Eds.), Flexible Automation and Intelligent Manufacturing: Establishing Bridges for More Sustainable Manufacturing Systems: Volume 2, Springer Nature Switzerland, Cham, 2024, pp. 187–199. https://doi.org/10.1007/978-3-031-38165-2_23.

[15] A. Löcklin, H. Vietz, D. White, T. Ruppert, N. Jazdi, M. Weyrich, Data administration shell for data-science-driven development, Procedia CIRP 100 (2021) 115–120. https://doi.org/10.1016/j.procir.2021.05.019.

[16] M. Pourjafarian, C. Plociennik, M.H. Rimaz, P. Stein, M. Vogelgesang, C. Li, S. Knetsch, S. Bergweiler, M. Ruskowski, A Multi-Stakeholder Digital Product Passport Based on the Asset Administration Shell, in: 2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA), IEEE, 2023. https://doi.org/10.1109/ETFA54631.2023.10275715.

[17] M.A. Inigo, A. Porto, B. Kremer, A. Perez, F. Larrinaga, J. Cuenca, Towards an Asset Administration Shell scenario: a use case for interoperability and standardization in Industry 4.0, in: P. Varga, D. Zuckerman (Eds.), NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium, IEEE, 2020. https://doi.org/10.1109/NOMS47738.2020.9110410.

[18] X. Ye, J. Jiang, C. Lee, N. Kim, M. Yu, S.H. Hong, Toward the Plug-and-Produce Capability for Industry 4.0: An Asset Administration Shell Approach, IEEE Industrial Electronics Magazine 14 (2020) 146–157. https://doi.org/10.1109/MIE.2020.3010492.

[19] A. Wagner, M. Volkmann, J. Hermann, M. Ruskowski, Machining of Individualized Milled Parts in a Skill-Based Production Environment, in: F.J.G. Silva, A.B. Pereira, R.D.S.G. Campilho (Eds.), Flexible Automation and Intelligent Manufacturing: Establishing Bridges for More Sustainable Manufacturing Systems: Volume 1, Springer Nature Switzerland, Cham, 2024, pp. 283–292. https://doi.org/10.1007/978-3-031-38241-3_32.

[20] Z. Muller-Zhang, T. Kuhn, A Digital Twin-based Approach Performing Integrated Process Planning and Scheduling for Service-based Production, in: F. Gao (Ed.), 2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA), IEEE, 2022. https://doi.org/10.1109/ETFA52439.2022.9921643.

[21] A. Alexopoulos, G. Kalogeras, K. Koutras, A. Kalogeras, Why Asset Administration Shells: A Survey on Uses and Challenges, IEEE Access 13 (2025) 126582–126609. https://doi.org/10.1109/ACCESS.2025.3589931.

[22] S. Wurm, V. Lohrmann, M. Wieczorek, P. Blanke, C. Fimmers, O. Petrovic, W. Herfs, Service-based tool lifecycle analysis based on AAS, Procedia CIRP 130 (2024) 1562–1568. https://doi.org/10.1016/j.procir.2024.10.283.

[23] F. Kaya, E. Şanlı, Ö. Albayrak, P. Ünal, P. Kirci, Asset Administration Shell Tool Comparison: A Case Study with Real Digital Twins Used in Petrochemical Industry, Sensors (Basel) 25 (2025). https://doi.org/10.3390/s25071978.

[24] A. Mazur, H. Peters, A. Artelt, L. Koller, C. Hartmann, A. Trächtler, B. Hammer, Studying the Generalization Behavior of Surrogate Models for Punch-Bending by Generating Plausible Counterfactuals, in: W. Senn, M. Sanguineti, A. Saudargiene, I.V. Tetko, A.E.P. Villa, V. Jirsa, Y. Bengio (Eds.), Artificial Neural Networks and Machine Learning – ICANN 2025, Springer Nature Switzerland, Cham, 2025, pp. 192–203. https://doi.org/10.1007/978-3-032-04555-3_16.

[25] H. Wang, abqpy 2025: Type hints for Abaqus/Python scripting, 2025. https://github.com/haiiliin/abqpy (accessed 16 January 2026).

[26] Industrial Digital Twin Association, IDTA 02008-1-1 Time Series Data (2023). https://github.com/admin-shell-io/submodel-templates/blob/main/published/Time%20Series%20Data/1/1/IDTA%2002008-1-1_Submodel_TimeSeriesData.pdf.