

We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

5,300

Open access books available

130,000

International authors and editors

155M

Downloads

Our authors are among the

154

Countries delivered to

TOP 1%

most cited scientists

12.2%

Contributors from top 500 universities



WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.

For more information visit www.intechopen.com



Chapter

Operationalizing Heterogeneous Data-Driven Process Models for Various Industrial Sectors through Microservice-Oriented Cloud-Based Architecture

Valdemar Lipenko, Sebastian Nigl, Andreas Roither-Voigt and Zelenay David

Abstract

Industrial performance optimization increasingly makes the use of various analytical data-driven models. In this context, modern machine learning capabilities to predict future production quality outcomes, model predictive control to better account for complex multivariable environments of process industry, Bayesian Networks enabling improved decision support systems for diagnostics and fault detection are some of the main examples to be named. The key challenge is to integrate these highly heterogeneous models in a holistic system, which would also be suitable for applications from the most different industries. Core elements of the underlying solution architecture constitute highly decoupled model microservices, ensuring the creation of largely customizable model runtime environments. Deployment of isolated user-space instances, called containers, further extends the overall possibilities to integrate heterogeneous models. Strong requirements on high availability, scalability, and security are satisfied through the application of cloud-based services. Tieto successfully applied the outlined approach during the participation in FUTURE DIRECTIONS FOR PROCESS INDUSTRY OPTIMIZATION (FUDIPO), a project funded by the European Commission under the H2020 program, SPIRE-02-2016.

Keywords: industrial optimization, model predictive control integration, machine learning model integration, Bayesian network integration, enterprise resource planning (ERP) forecast model integration, prediction model integration, model calculation graph, microservice-oriented architecture, cloud computing

1. Introduction

In the area of industrial manufacturing performance optimization, prediction models are often used to predict future plant outputs in order to increase product quality and energy efficiency or optimize planning of plant maintenance activities.

Forecasts and predictions are utilized to generate lead time to plan and operate the manufacturing processes in a highly optimized way. To achieve predictive

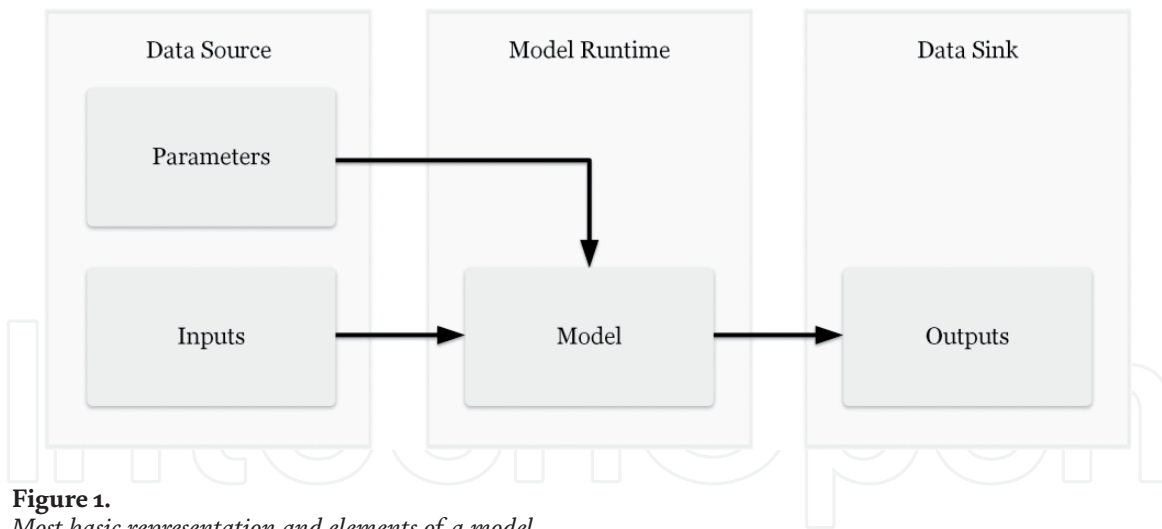


Figure 1.
Most basic representation and elements of a model.

operation on optimized performance, analytical calculation models are developed and operationalized to calculate the future behavior of manufacturing plants. This chapter describes the techniques and methods for operationalizing the prediction models in industrial manufacturing environments.

A model is defined as a simplified mathematical representation of a real natural process. Such process can be an industrial plant (e.g., a heat exchanger) or a complex plant like a continuous production machine (e.g., large-scale waste incineration steam boiler). Models are also called digital twins in applications where a model represents the real plant in a very high degree of details as in finite-element-method (FEM) model of a system (machine) or its components.

A model in the context of this chapter is characterized by three main features:

1. The model is a representation of a real plant.
2. The model is simplified “version” of the real plant that does not include all properties or behavior.
3. The models can be generalized, so they represent a type or a class of a real plant and not necessarily a real existing instance of a plant at a specific time.

Models are typically operationalized calculation (software) modules that allow a causal calculation of outputs from inputs and parameters. A model is defined as a calculation module that can predict the future industrial plant behavior. Such “plant behavior” can be the quality of produced product and the consumption of energy such as steam, electricity, raw material, and chemicals. Operating industrial plants based on predictive analytics allows optimized planning and real-time optimization. An industrial manufacturing or process industry plant has usually strictly separated information technology (IT) and operational technology (OT) systems (**Figure 1**).

2. Requirements and convergence of industrial system architecture

Industrial manufacturing industry continuously seeks for performance optimization strategies and ways of operationalizing new methods for improving product quality, production efficiency, energy efficiency, emission reduction, and, of course, cost reduction techniques. Digital transformation programs need to support these

performance optimization goals. By driving convergence of OT and IT on all levels of enterprises (people, processes, interfaces, and system architecture), beneficial effects from integrated and consistent data and information utilization can be achieved.

Furthermore, a state-of-the-art operational architecture should be open and scalable for a whole ecosystem of internal or external partners to benefit from a learning culture on how to operate all production and business processes at maximum performance levels under changing market, raw material, and environment conditions (**Figure 2**).

Typical areas of industrial OT systems are as follows:

- machines and parts of machines (pumps, mixers, valves, tanks, etc.);
- automation systems [distributed control system (DCS), programmable logical controller (PLC), supervisory control and data acquisition (SCADA) system]; and
- operator user interfaces of SCADA system and DCS.

Typical areas of industrial IT systems are as follows:

- business applications like customer relationship management (CRM) system;
- business intelligence and data warehousing;
- enterprise resource planning (ERP) system;
- data analysis;
- computing systems and technology; and
- data warehouse and storage systems.

The main questions relevant in the design process of an industrial prediction model operationalizing framework development can be as follows:

- How to make exchange of prediction models as easy as possible for an industrial ecosystem with partners from process industry and scientific community and commercial partners?
- What framework elements have impact on the calculation performance of large-scale models and what are the performance requirements based on the dynamic behavior of the processes of interest?
- What architectural security elements are required to ensure a safe operation of the model calculation runtime system and user interfaces.

2.1 Requirements by heterogeneous industries

Various industries have significantly different requirements for the integration of prediction and optimization models.

Examples of different industries that benefit from prediction/forecast model integration are shown in **Table 1**:

Industry category	Model value/benefits
Continuous process industry (e.g., pulp and paper mill, continuous waste water treatment plant, oil and gas refinery)	<ul style="list-style-type: none"> • Production rate of continuous production machines • Tank level utilization for stable production • Optimized quality stability
Original equipment manufacturer (OEM), e.g., micro-combined-heat-power unit manufacturer	<ul style="list-style-type: none"> • Improved overall equipment efficiency • Machine flexibility in conditions with changing raw material properties • Enable predictive maintenance use cases
Energy and utilities (e.g., waste incineration power plant, district heating network)	<ul style="list-style-type: none"> • Increased process stability (temperatures, pressure, steam flow) from predictive boiler operation using feedforward model predictive control • Increased energy efficiency by predictive control of excess combustion air (=O₂) control
Discrete manufacturing (e.g., automotive)	<ul style="list-style-type: none"> • Increased overall equipment efficiency (productivity time quality acceptance ratio) • Reduced costs due to lower reject rates

Table 1. Industry categories and model benefits (by examples).

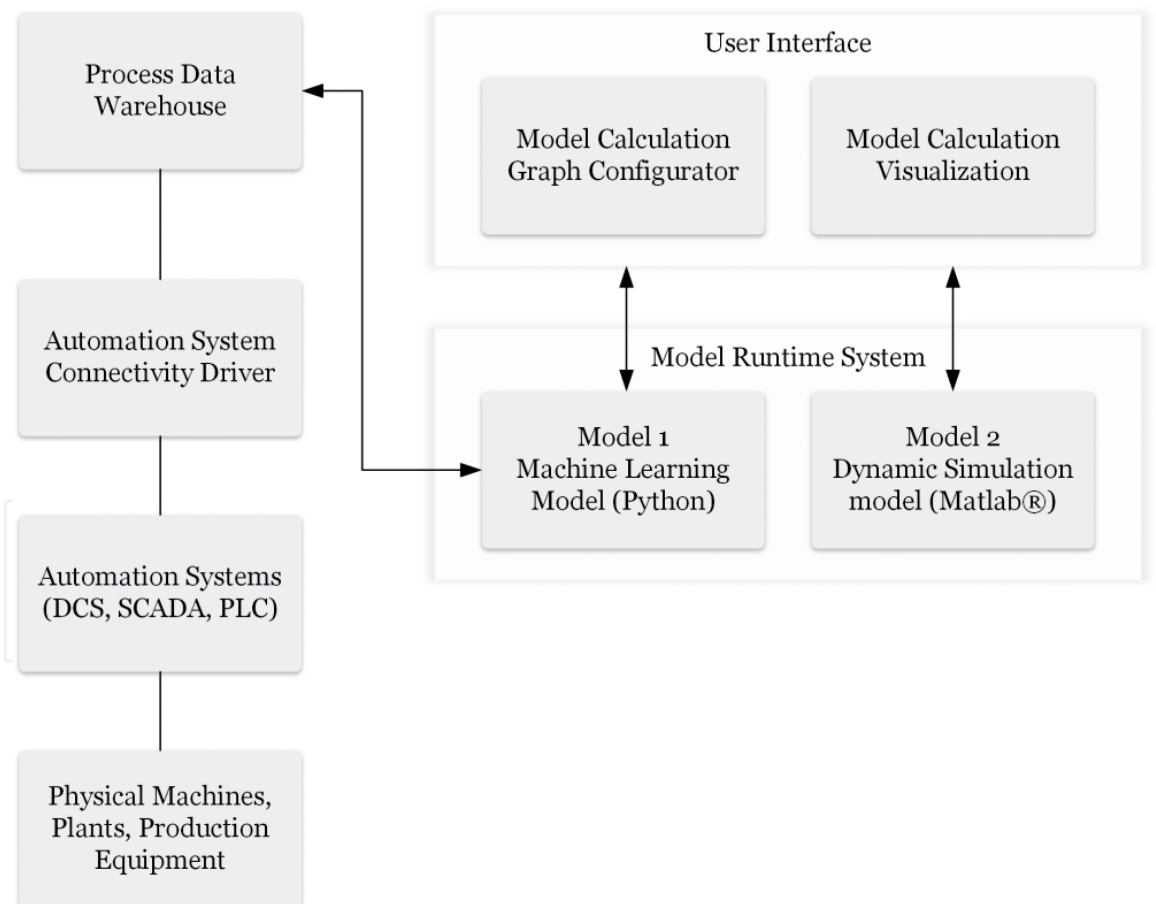


Figure 2. Model operationalization framework overview (examples of).

2.2 Requirements by heterogeneous models

Forecast and prediction models can be of various types and purposes of integrated usage by a smart manufacturing system. The huge spectrum of very specific requirements makes it hard for practitioners to find models from existing

libraries that fulfill the requirements of the specific application. An ontology-based approach to classify and identify such model application usage scenarios can be found in Ref. [1]. The challenge of model exchange can be solved by using standards for model exchange such as functional mockup interface (FMI) or functional mockup units (FMU), as specified in Ref. [2].

Model examples illustrating the range of requirements are as follows:

- prediction of future process information based on production plan;
- prediction of future process output based on (just) recent process inputs, process states, and predicted future disturbance variables;
- prediction models for use in model predictive control (MPC). Dynamic MPC models can be linear or nonlinear; and
- first principle physical models (dynamic, static).

Models can be developed in various simulation software tools and have very different requirements when it comes to the runtime environments. As models are software components, many dependencies need to be fulfilled to run the models. Running models in this context means execution with actual inputs and user-specified parameters to calculate the predictions for a specific period (usually a time period in the future).

In order to avoid a complicated or even contractionary model runtime system architecture, a containerization technology can be used. In a software container (like Docker [3]), each model gets the required dependencies installed in the container rather than on the global runtime system. Utilizing container technologies

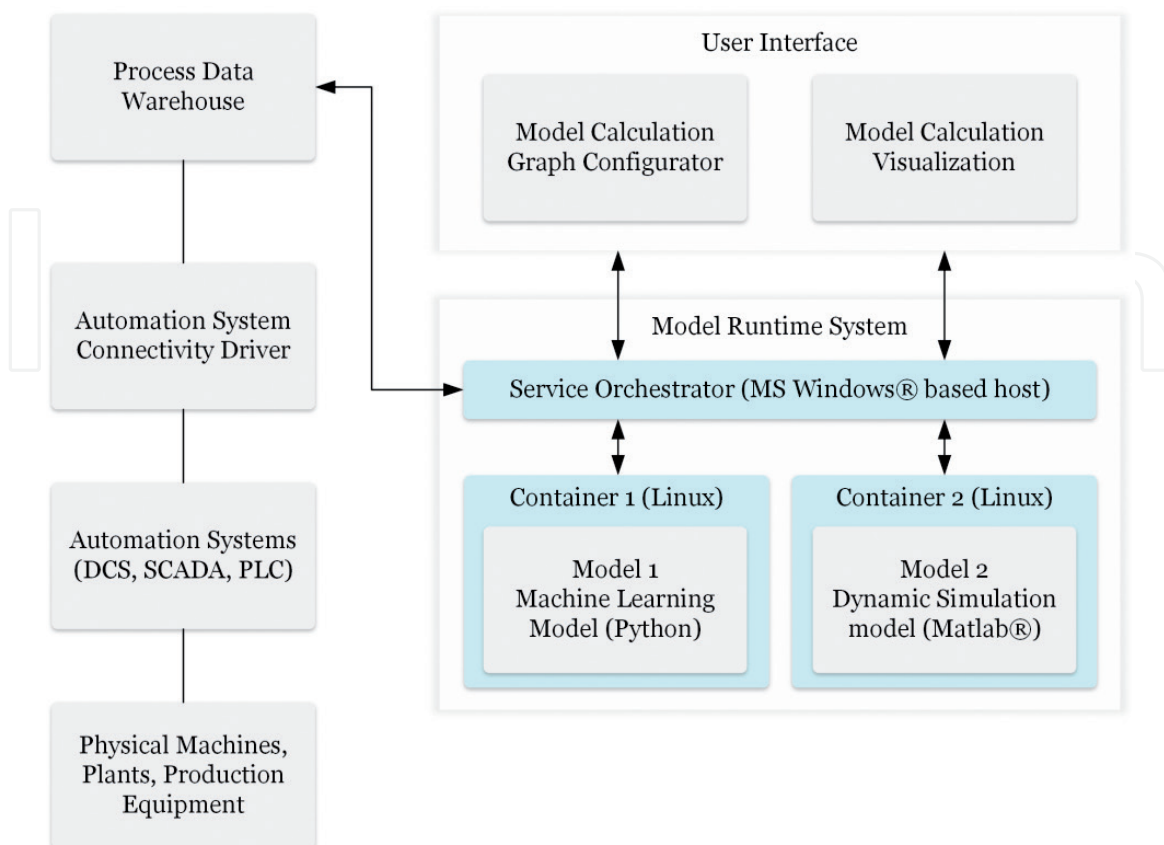


Figure 3.
Microservice-based architecture for operationalized model integration.

allows a clean deployment of operational models in their own containerized runtime system avoiding any additional requirements on the main application level as a container provides all required dependencies.

The resulting system architecture is called microservice architecture as it contains smaller independent software components packed into smaller application units (containers). An additional service orchestrator application is required to handle the messages between the OT systems and the microservices (containers) and the user interface.

Figure 3 shows the system architecture for an example with two containers where Linux is used as container operating system and the model container host server can be Microsoft Windows[®] based. This scenario allows the integration of models for all commonly used operating systems and can therefore provide all dependencies of other software components needed by the models to run/execute properly.

3. Calculation graph configurator

This section will cover why it is beneficial to use a calculation graph configurator when integrating one or more models into one software solution. Furthermore, it will be explained how node-RED can be used as a calculation graph configurator [4].

3.1 Calculation graph configurator overview

Since executing models often requires multiple calculation steps, for example, preprocessing the model input data or apply filters when selecting input data, a visual tool to connect and modify these calculation steps is beneficial. For example, in the FUDIPO project Node-RED, a flow-based programming tool is used as a calculation graph configurator. One big advantage of a tool like Node-RED as a calculation graph configurator, is that it saves a lot of cost due to the fact that Node-RED is highly customizable and open source. Therefore developing a custom calculation graph calculator, is not necessary. Furthermore, Node-RED is cross platform compatible as it runs on Node.js. There is also an official Docker image for Node-RED [4].

3.2 Node-RED as a calculation graph configurator

In Node-RED, calculation steps are called nodes. The nodes communicate via JavaScript Object Notation (JSON) messages. Node-RED provides a base set of nodes with a special functionality like nodes making a HTTP-Request or executing a JavaScript code. In addition to the base nodes, the community develops and contributes nodes that are available, thus providing very versatile functionalities. Node-RED also offers a dashboard where the model results could be displayed, although the default dashboard of Node-RED does not support multiple users, it should just be used for debugging or if only one user accesses the user interface (UI). Another great feature of Node-RED is that it provides a node to map data. Data mapping is required when the data source variables and the model input variables are not in the same structure. In some cases, the data must be mapped twice, once before the model execution, and a second time after the model has been executed to store the model results, as shown in **Figure 4**.

Node-RED's dashboard offers some basic UI elements, like charts, input forms, buttons, switches, and slides; however, it is also possible to write HTML, cascading style sheet (CSS), or JavaScript code directly, and this offers the ability

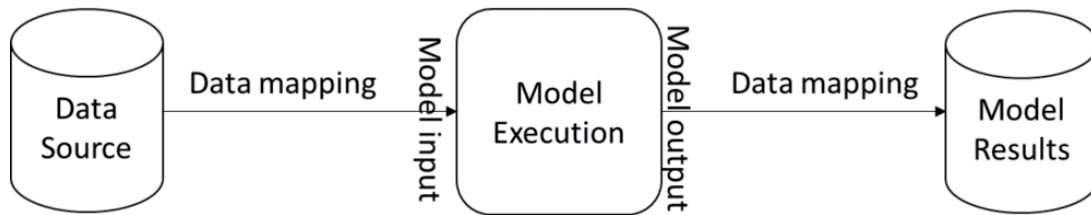


Figure 4.
Data mapping for normalized model execution.

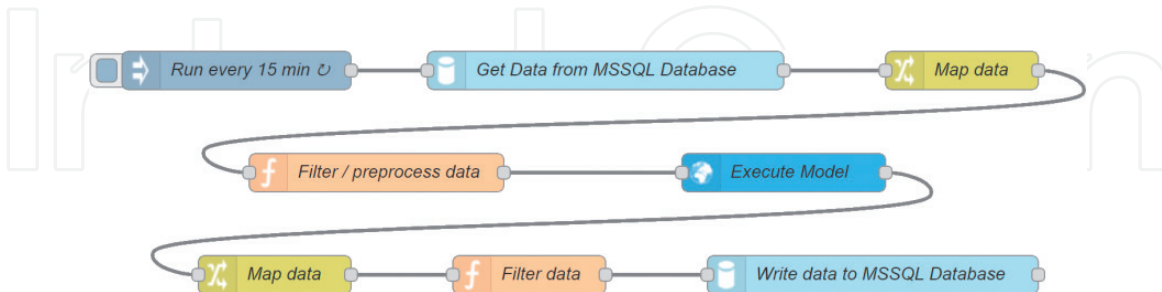


Figure 5.
Calculation graph example in Node-RED.

to create a highly customizable UI within Node-RED. Furthermore, Node-RED offers the possibility to customize its theme to match the company's color scheme. Additionally, Node-RED also offers the ability to secure the editor and dashboard user interface with a username and password, so if you have sensible data or do not want your Node-RED to be publicly available, it can be secured with a login mask [4].

3.3 Node-RED calculation graph example

An example calculation graph would be to retrieve data from an MicroSoft Structured Query Language (MSSQL) Server → map the data to match the model input data → filter or preprocess the input data → execute the model → map data back to match the schema of the MSSQL database → filter the model result data → save the filtered model results in a MSSQL server. In **Figure 5**, the example just explained is realized in Node-RED [4].

The first node runs the calculation graph every 15 minutes. At the time of writing, the interval can range from 1 second to 596 hours. The "Get Data from MSSQL Database" is a contribution node, which means that a user created this node and provided the source code, so everyone can use it. This node retrieves data from an MSSQL Server using Structured Query Language (SQL) statements. The yellow "Map data" node uses JSONata to change the structure of a JSON object. In the "Filter/preprocess data," node JavaScript is used to apply custom-made filter and preprocessing algorithms. The "Execute Model" in this example sends a HTTP POST request to an Application Programming Interface (API), where the model is being executed and returns the model results as an HTTP response. The next node maps the model result data to the schema of the MSSQL database. Afterward, these data are filtered again before writing it back to the MSSQL database in the last node. An advantage of this solution is that there is only little effort needed to integrate models into the solution. The only thing that needs to be developed is API that executes the model. Node-RED also offers the possibility to execute terminal commands, so if a model can be executed via the terminal, even less effort is needed to integrate the model. This calculation graph could also be used for multiple systems or machines if the model is designed to do so [4].

4. User interface

In this section, it will be discussed how a user interface for displaying the model results and for uploading or tuning the models could be developed. The key features, a user interface should have, are as follows:

- visualizing model results with charts;
- allowing the model developer to upload a new version of the model; and
- allowing the model developer to test and tune the model.

Node-RED allows the model developers to tune and test their models with the live data; however, Node-RED does not offer the feature to upload new models. A possible solution for this would be to use Node-RED's dashboard to visualize the model results, to use Node-RED's editor to make it possible for the model developer to verify and tune his model(s) with live data, and to build a separate website enabling the model developers to upload newer and improved versions of their models. As already mentioned, Node-RED's dashboard has the big advantage that it is very easy to use, even people with little to no knowledge about HTML, CSS, or JavaScript can make a simple user interface in Node-RED. However, the major flaw in the provided dashboard is that it does not support multiple users; for small use cases or debugging purposes, this might be enough, but in large-scale applications where multiple people are going to use the dashboard, another solution must be used. In such cases, it is either possible to develop a new website and access the model results either from an API or from a database, or use a fork of Node-RED's dashboard, developed by the community, that supports multiple users [4].

Another possible solution would be to display the model results in the company's current software solution. If there is already a software that acts as a user interface for the machine data, it might be possible to display the model results too.

If the goal is to continuously integrate multiple models, a possible solution would be to develop a website where the model developers are able to upload and maybe even verify, possibly with real live data from the machines, their models. Even though such an automated process might be convenient, it requires a lot of development effort and it should be considered if the initial effort is worth spending. Another possibility is to initially integrate the model manually in the solution and provide the model developers the possibility to update their models, by uploading a new version to a website. This is especially useful if there are models that require frequent updates. For example, training a machine learning model requires a lot of computing power, due to this it is suboptimal to train the model on the server where the model is executed. Thus, training the model on a separate computer or server that is suited for such high loads is better; therefore, a web interface could be used to upload the newly trained model, providing the model needs to be trained continuously. Another possible feature such a website could have would be allowing model developers to test the newly uploaded model with live data. So, model developers are able to ensure that their models are working correctly.

To be able to display the model results in a website in most cases, a charting library is needed. One of the many JavaScript charting libraries is Highcharts[®]. The advantage of Highcharts[®] is that it is very configurable, well documented, and feature rich. Highcharts[®], for example, is able to export the data that are displayed in the chart as an Excel sheet or download the chart as an image. Highcharts[®] has many feature add-ons, called modules. Another big advantage of Highcharts[®] is

that it has a boost module, which significantly boosts the performance of the charts making it possible to display well over 500,000 datapoints in one chart while keeping about the same performance as a chart with 1000 datapoints [5].

5. Pulp and paper use case

Processes in the pulp and paper industry are considerably complex with significant time delays (up to few hours) resulting in major difficulties for appropriate process optimization and control. An example of such process is continuous cooking in pulp digester, aiming at removing lignin from wood chips [6]. The most widely used index for measuring residual lignin present in the pulp is kappa number [6]. The digester primary control objective consists in minimizing the variability of kappa number, keeping it in a small range within few percent of target value (too low and too high kappa numbers negatively both impact quality and production stability).

The current situation is shown in **Figure 6**.

Through utilization of various process-specific analytical data-driven models, it is possible to substantially improve the process control, hence also the product quality and production stability.

One of such data-driven approaches is to conduct the forecast of future observations, such as based on certain characteristics of wood chips to predict the resulting kappa number (which is otherwise known only with the delay of ca. 4 hours from the time when respective wood chips came into cooking process). Measurement of wood chips with near-infrared (NIR) provides the capabilities to extract data on lignin content, moisture, and reactivity of wood chips. Subsequent utilization of specific physical and/or machine learning models enables to forecast the resulting kappa number.

MPC integration is another important approach in this context, in particular accounting for multivariable specifics of the underlying process control. The presence of measurement noise and various complex chemical process uncertainties, which are common in the pulp and paper industry, can also effectively be addressed through MPC [7]. Here, data from NIR-based soft sensors coupled with various dynamic process models are the main enablers of a feedforward MPC [7].

Process diagnostics (such as identification of digester hang-ups, screen clogging, and channeling of liquor inside the digester [8]) can effectively be done by input of certain information on current system status to a causality tree, such as Bayesian Networks (BN), with subsequent inference for computations of probability of different faults. Hence, the resulting decision support system is a valuable tool to improve diagnostics and fault detection capabilities in pulp and paper applications.

A comprehensive view on model-based control and diagnostics for pulp digester provides [9] the possibility of feedforwarding the lignin content of incoming wood chips based on NIR measurement under the incorporation of modeling and simulation studies. Additionally, Rahman et al. [9] proposed a simple Bayesian network-based diagnostic approach to detect pulp digester faults. Rahman et al. [7] provided an approach for feedforward MPC as applied to the pulp and paper industry.

Moreover, the analytical data-driven models need continuous updating to utilize process changes and to learn from experience. The feedback can be done automatically, by process operators, maintenance staff, and so on.

Thus, **Figure 7** provides an overview of the potential process improvement as compared to the current situation.

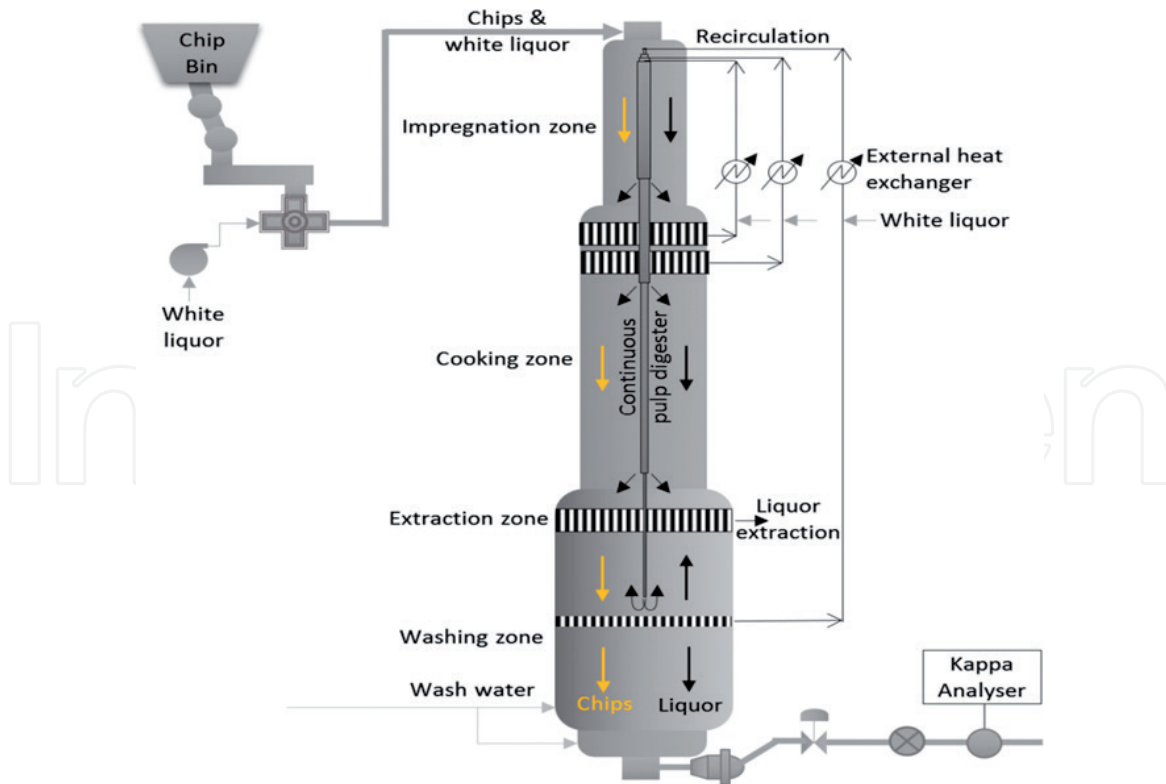


Figure 6. Pulp digester [8].

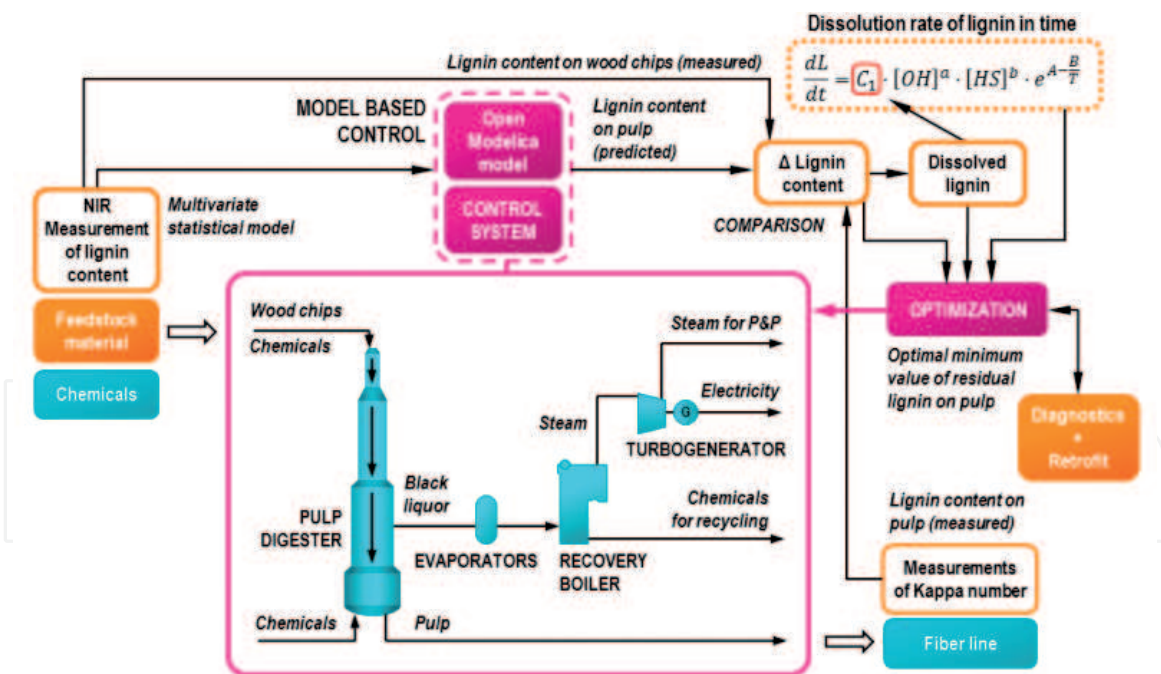


Figure 7. Pulp digester improved process [8].

6. Micro turbine powered combined cycle

A key factor to reduce the operating cost of a micro gas turbine (MGT) powered combined heat and power fleets is to improve the maintenance strategy. The goal is to stop maintaining the micro gas turbines on a timely basis, instead maintaining MGT based on anomaly detection (condition-based maintenance) with specifically developed models that are able to detect MGT faults. In addition to lowering

maintenance and operating costs using such models can also increase the safety of micro gas turbines. Two ways to develop such models would be either by using a physics-based approach or by using a data-driven approach, and the best results are achieved when combining a physics-based model and a data-driven model. The challenge when developing the models for a fleet of micro gas turbines is that often models are tuned for a specific machine and cannot be applied to a similar machine; however, when having a fleet of thousands of micro gas turbines, it would require way too much effort to develop a model for each system; therefore, the model must detect anomalies for multiple MGTs. To sum it up models should predict the maintenance actions long before the MGT fails, this will likely increase the safety, reduce the maintenance cost, and possibly increase the availability of MGTs [14, 15].

Physics-based models are based on mathematical formulas and constraints between sensor data. A physical model for MGTs basically simulates a micro gas turbine and compares the simulated values with the real sensor data. With the results, a degradation of a MGT can be calculated. The main disadvantage of such models is that they require an expert knowledge of the machines they are developed for. In contrast, data-driven models do not require such comprehensive knowledge of the field. Data-driven models use big quantities of sensor data and known failures. These data-driven models are able to classify and predict the future failures based on learning (machine learning) from the historical data. The drawback of data-driven models is that it is often not comprehensible how the model results are exactly calculated.

6.1 Example system architecture for micro gas turbines

This subsection will illustrate an example system architecture for micro gas turbine powered heat and power fleets.

Figure 8 represents an example system architecture for MGT fleets. Every MGT in the fleet writes its sensor data to a database, which is in the Azure [16] cloud for maximum scalability and availability. The SQL server has two databases, one for the sensor data and a separate database for the model results. The databases are separated because if one database crashes, the other will keep running. If even more reliability is needed, the two databases can also be in separate SQL servers. The virtual

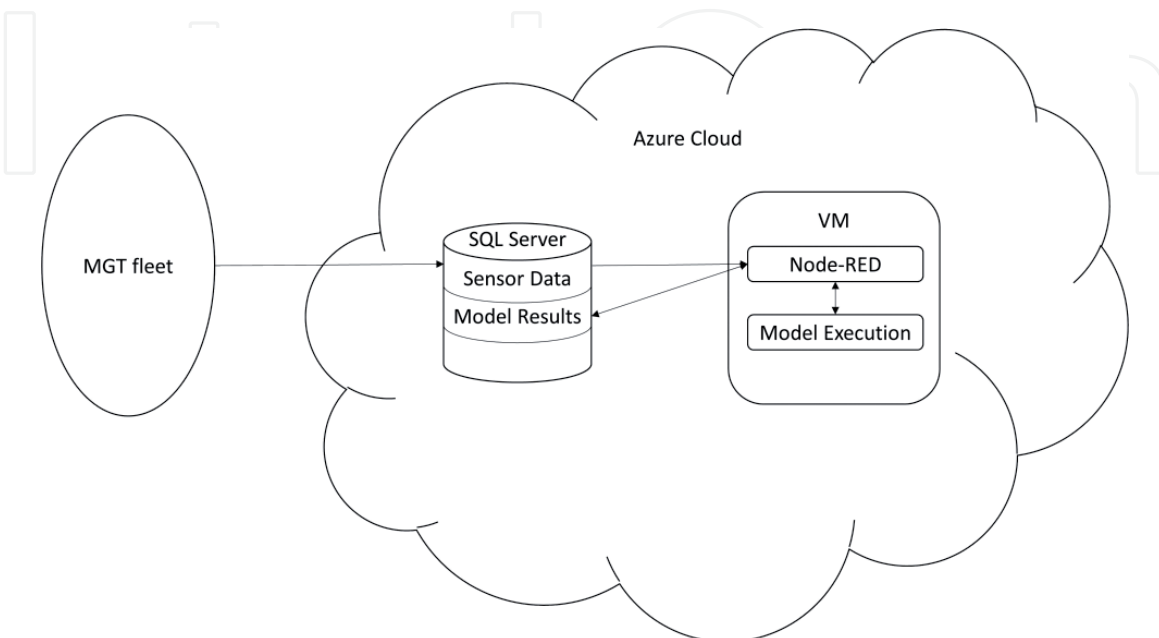


Figure 8.
Example system architecture for MGT fleets.

machine (VM) runs Node-RED as a calculation graph configurator and allows the model developers to verify and tune their models. Node-RED then executes the calculation graph as explained in Section 3.3. In this example, Node-RED's dashboard is being used to validate, tune, and debug the models. Additionally, the model results (e.g., future maintenance actions) are visualized in the company's own UI. Furthermore, the model results and sensor data could be processed and displayed in the UI of the customer, so the owners of the MGT(s) could see how good their MGT(s) perform, or even allowing them to plan for the future maintenance costs [4, 5].

7. System environment

In the present setting, the view on the overall system architecture may not be limited to IT aspects but is necessarily extended to OT. Hahn [10] provides an in-depth analysis of the differences between IT and OT in the domain of industrial control systems (ICSs). On the one hand, OT is traditionally seen as the heart of ICS. On the other hand, modern ICS increasingly utilizes IT capabilities, creating a convergence of OT and IT domains [10]. For an in-depth understanding of the impact and posed challenges of OT and IT consolidation, it is therefore important to align on different principles to handle data in these both domains, together with differences in how the overall systems are operated and managed, which technologies are used to support them, and so on. Hahn [10] provides the respective explanations as follows:

- **Operational domain.** In order to control and monitor physical processes (such as power grids, pulp production, and oil refinery), various sensors, actuators, controllers, and human operators are altogether utilized in ICS. The resulting unique operational requirements are much different from the traditional IT environments, whose focus is more strongly on controlling and managing the data, retrieved here from the underlying OT.
- **Technical domain.** The unique technical requirements for the software used to support the operations of ICS mainly result from or are closely related to specific communication protocols and architectures; real-time performance demand; domain-specific device manufacturers and integrators; complex integration of digital, analog, and mechanical controls; and so on.
- **Managerial domain.** From the management of OT system's point of view, their underlying complex physical infrastructure usually requires much larger capital investments than it is the case for the IT systems. Hence, the operation of ICS is subsequently usually planned for decades, in order to recuperate the cost.

Thus, in order to successfully integrate specific solutions from the IT domain (in particular, analytical data-driven solutions) with OT in a holistic ICS, the following requirements should be, respectively, satisfied [11]:

1. **Data source requirements.** Support for data produced by various industrial machines and sensors, usually connected in one network, without the possibility to access over a common application programming interface (API) is crucially important. The unique characteristics of single heterogeneous elements should be harmonizable for further processing.

2. Processing requirements. Data from sensor instrumented industrial devices commonly possess typical Big Data characteristics, so-called four Vs:

- *velocity*: data arrive rapidly from many different sources;
- *volume*: huge amounts of data produced in seconds or milliseconds;
- *variety*: structured, semi-structured, and unstructured data appear; and
- *veracity*: occurrence of noisy or otherwise of a poor-quality data.

This makes such data very challenging to manage, integrate, and analyze.

3. Human interface requirements. The insightful results from assessed data are necessary to be made available to operations professionals in convenient manner for them. In other words, the respective process database should eventually contain all the data required to accordingly inform process responsible (in form of visualizations, notifications, alerts, etc.).

4. Security requirements. Integration, development, deployment, extension, and customization are required to take place in secure environments with ensured security standards on at least the same level as already in-place.

An important extension to the considerations regarding IT and OT is the runtime environment itself, or more precisely the decision between installing and running single solution components on the premises of demonstrators (on-prem) as opposed to remote facilities (cloud-based). The outlines of advantages and disadvantages of on-prem and cloud-based solutions are, respectively, depicted in **Tables 1–3**. At the same time, since in this case the single advantages and disadvantages are often overlapping, the summarized information should be seen as a very general overview only.

Since on-prem and cloud-based solutions have own strong advantages and disadvantages, there is naturally no single best choice for every different demonstrator. In other words, it is crucially important to enable an efficient system architecture setup for on-prem the same as cloud based. Then, the decision for a particular setup is left to be met solely in accordance with respective internal policies, with posing

Advantages of on-prem solutions	Disadvantages of on-prem solutions
Great possibilities for customizations	Solution implementation takes longer
Possibility for specific security policies	High system maintenance effort
High solution design flexibility	High system management cost
In-house system and data knowledge	High upfront investment

Table 2.
Advantages and disadvantages of on-prem solutions.

Advantages of cloud-based solutions	Disadvantages of cloud-based solutions
Short implementation time	Limited customization possibilities
No need for system maintenance	Security depends on cloud provider
Low upfront investment	Low solution design flexibility

Table 3.
Advantages and disadvantages of cloud-based solutions.

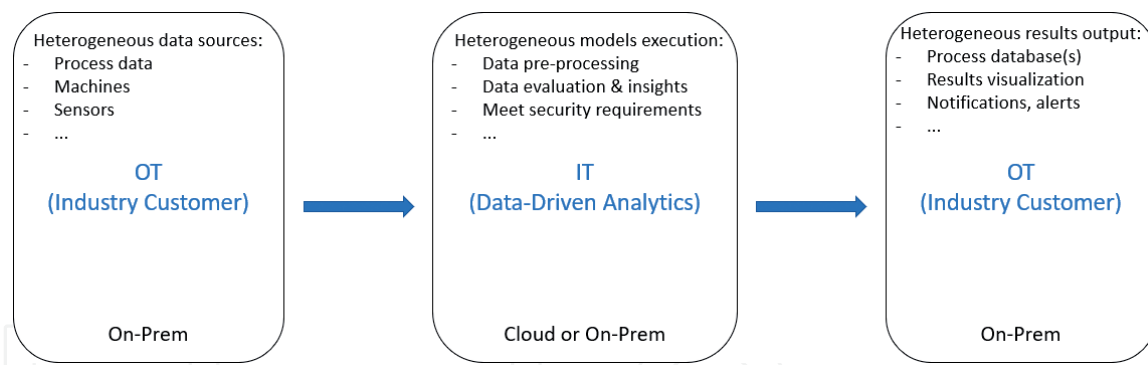


Figure 9.
Simplified generic system overview.

no technological limitations whatsoever. Hence, disregarding the differences between the outlined system architectures, they can be consolidated as depicted in **Figure 9**.

Hence, the consolidated system environment may easily be generalized as of a cloud-based nature, differencing solely in certain administrative responsibilities of the underlying technical infrastructure.

8. Microservice architecture

The decision in favor of microservice architecture (MSA), which is a variation of service-oriented architecture (SOA), naturally results from the cloud-based (either a public cloud or an on-prem configuration) specifics of the underlying solution. In a nutshell, a solution adopting microservice-based architecture consists of a large number of small services, each responsible for a single-specific aspect (e.g., data access, execution of certain model, and specific data preprocessing step) [12]. Hence, the main benefits of solutions implemented based on MSA include their major scalability capabilities (both scaling up and scaling down depending on the present circumstances), reusability, loose coupling, and their advanced technology agnostic nature, which are probably the most important in the discussed context. The latter advantage of MSA enables easily utilize different runtime environments and programming languages in single microservices of one mutual solution, hence also adapting to technological changes to avoid technology lock-in, and so on [12].

Coming back to the previously elaborated requirements in the context of OT consolidated with IT to successfully implement analytical data-driven solutions, MSA solves these as follows:

1. Data requirements. The heterogeneous nature of the underlying data produced by industrial machines and sensors is best supported through the creation of separate microservices responsible for particular types of data, retrieving data from specific sources, and so on. Moreover, every new type of data or data source can further be easily supported by simply adding a new and independent respective microservice to the overall solution. Through further microservice, heterogeneous data can be harmonized and brought to common format and structure for the subsequent processing stages.
2. Processing requirements. The architecturally unlimited number of microservices in-place best support the requirement to process even data produced in near real time through simply extending the number of instances of respective

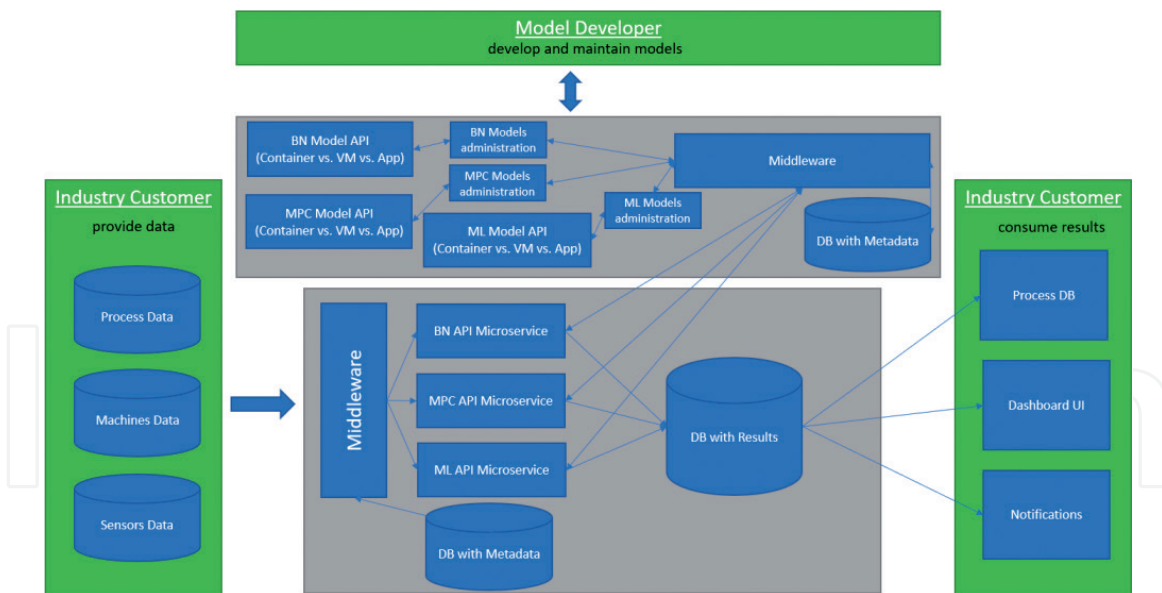


Figure 10.
 Microservice-based system outline.

microservices. The only bottleneck could be imposed through requests dispatching entity, such as a certain API management microservice, which is nevertheless efficiently replicable nowadays as well. Thus, all of the V's of Big Data can efficiently be approached by utilizing MSA.

3. Human interface requirements. The way of how results of data processing and analysis are made available to process professionals (process operators, process engineers, etc.) is again not limited by the architecture and only requires the implementation of appropriate single-independent microservices for writing the data to respective type of process database, or even simply visualizing in-place in the underlying solution. Moreover, the initially chosen implementation is easily adjustable during the whole solution lifecycle based on, for example, specific usability requirements.
4. Security requirements. Single microservices made responsible for security act as respective middleware, thus ensuring conformity to require security standards. Again, customization and adaptability are major benefits of choosing MSA as opposed to monolithic solutions.

An exemplarily overview of how MSA can successfully be utilized for operationalizing the heterogeneous data-driven process models is depicted in **Figure 10**.

9. ERP integration of prediction models

In today's times, the ERP system has a central role in almost every company. Its capabilities help the business to act, plan, and decide on the data, which will be collected in such systems. In all firms, such decision-making processes are happening every day. The question is how a company could become more competitive than its competitors. What if, company leaders could predict future events or would be able to connect the dots beforehand. This and maybe other reasons are the motivation to integrate the prediction models in ERP systems.

The challenge is, to bring the best parts of business know-how, software engineering, and data science together, to be able to predict the future.

The first step is to understand the company's business. Only with that knowledge, problems can be identified, which are worth enough to be avoided to save leaders in firm's time and costs.

One example of such problem could be the question: When and how much will my customer order which products? There are too many parameters to answer this question with a deterministic approach, which means the only way to solve this problem is to get an empiric way of thinking. So, companies can work with that what they already gathered in so many ways, with data.

This leads to the second step to get the understanding of that data. The advantage of having an ERP system is that the information is already collected and described. To get one step further, the challenge is to find out or explore where these data are stored and verify the quality.

Once this is done, the content of the third step is to prepare the data for modeling. Simply spoken, the task here is to bring the data in a correct format that prediction models can work with. Every model needs a different format, which means in this step, it is also needed to know which model should be used. If the formatted data and prediction model fit together, the next step can be done.

Step 4 is about modeling. In this phase, data and the selected prediction model will be evaluated, tested, and improved in a couple of iterations or even exchanged. The quality of a prediction model can be measured in predictive power and predictive robustness. The power measures the capacity of the input variables to explain the target. The robustness is the ability to display the same level of performance on new data sets as training ones.

The final step is to deploy and integrate the prediction model in the ERP system. To show how this can be accomplished, the following paragraphs will rely on the ERP solution Systems, Applications, and Products (SAP).

A typical SAP on-premise installation always comes with an application server called SAP S/4HANA and a database management system called SAP HANA [13–16].

When and how much will my customer order which products?

In the paper business, the products can be distinguished by different characteristics, but one major characteristic is the grade. So, for every grade, different raw materials are needed to be ordered and purchased. Also, a major factor is that for every grade, the paper machine needs a different setup, which leads also to effort and costs. So, for the management of a paper producing company, it would be beneficial to predict the customer's behavior for the next month to save time and costs.

The information which customer orders which product with amount and point of time is maintained the so-called customer sales order. Information to the product is available in the material master. The following database tables are meant: VBAK (Sales Document Header), VBAP (Sales Document Position), MARA (Material Master), and A USP (characteristic values).

To be able to work with that data, it is needed to bring all tables in view to aggregate the information on a monthly basis. This can be accomplished with a so-called core-data-service (CDS), which is nothing else than a view with special functionalities. The CDS-View plays a major role when it comes to data preparation and analysis.

As there is the need to predict the future time events, the auto regressive integrated moving average (ARIMA) model will be chosen for this scenario. The ARIMA model is delivered within the HANA database by SAP Predictive Analytics Library (SAP PAL) and can be accessed via ABAP Managed Data Procedures (AMDP) from the application layer. It must be mentioned that there are also different integration scenarios possible. For example, a R- and/or Python-Integration can be implemented with SAP as well. This approach will be used for high sophisticated models where the functionalities of SAP PAL cannot match for this requirement.

To integrate this predictive function in existing applications or reporting tools, an Advanced Business Application Programming (ABAP) will be written for this purpose.

The ABAP executes the training of the model and the forecast for the next period at one point of time.

To get an overview how this solution with its different integration scenarios could like the following illustrations should help to get a better understanding what has been described before (**Figure 11**).

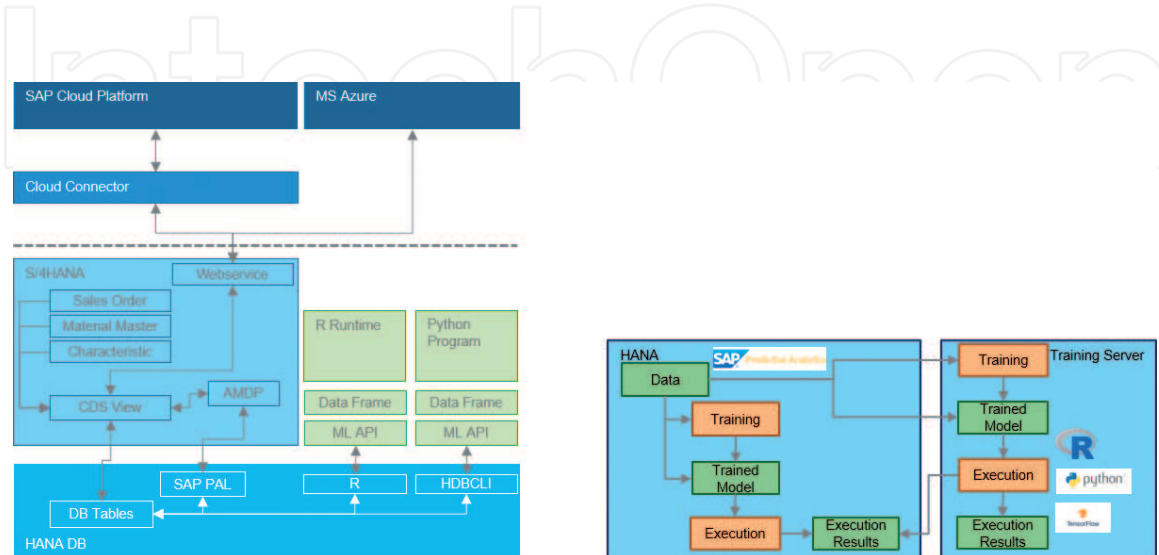


Figure 11.
Architecture for SAP ERP integration of predictive models.

10. Conclusions

Operationalizing and deployment of industry process prediction models can be achieved efficiently by setting up a microservice-based architecture that gives not only the industry companies but also the scientific community full flexibility in using software components to run the prediction models. Successful integration of heterogeneous models for different types of industries has shown high added value for operating the prediction models for business performance optimization. Process industry and OEMs benefit by the proposed architecture from a very fast and cost-efficient implementation of models into their OT and IT environment.

Acknowledgements

The work presented in this paper is carried out in connection to the FUTURE DIRECTIONS FOR PROCESS INDUSTRY OPTIMIZATION (FUDIPO) project. This project has received funding from the European Union's Horizon 2020 SPIRE-2 (Sustainable Process Industry Research) program under Grant Agreement No. 723523.



Conflict of interest

The authors declare no conflict of interest.

Abbreviations


ABAP	advanced business application programming
AMDP	ABAP managed data procedures
API	application programming interface
ARIMA	auto regressive integrated moving average
BN	Bayesian networks
CDS	core-data-service
CRM	customer relationship management
DCS	distributed control system
ERP	enterprise resource planning
FMI	functional mockup interface
FMU	functional mockup unit
FUDIPO	FUTURE DIRECTIONS FOR PROCESS INDUSTRY OPTIMIZATION
ICS	industrial control systems
IT	information technology
JSON	JavaScript object notation
MGT	micro gas turbine
MPC	model predictive control
MSA	micro service architecture
MSSQL	Microsoft structured query language
NIR	near-infrared
OEM	original equipment manufacturer
OT	operational technology
PLC	programmable logical controller
SAP	systems, applications, and products
SCADA	supervisory control and data acquisition
SOA	service-oriented architecture
SPIRE	sustainable process industry through resource and energy efficiency
SQL	structured query language
VM	virtual machine

Author details

Valdemar Lipenko, Sebastian Nigl, Andreas Roither-Voigt* and Zelenay David
Tieto Austria GmbH, Vienna, Austria

*Address all correspondence to: andreas.roither-voigt@tieto.com

IntechOpen

© 2020 The Author(s). Licensee IntechOpen. Distributed under the terms of the Creative Commons Attribution - NonCommercial 4.0 License (<https://creativecommons.org/licenses/by-nc/4.0/>), which permits use, distribution and reproduction for non-commercial purposes, provided the original is properly cited. 

References

- [1] Novák P, Šindelář R. Applications of ontologies for assembling simulation models of industrial systems. In: OTM Confederated International Conferences on the Move to Meaningful Internet Systems; 17 October 2011. Berlin, Heidelberg: Springer; 2011. pp. 148-157
- [2] FMI-Standard [Internet]. Available from text [Accessed: 13 August 2019]
- [3] Docker Container Technology [Internet]. Available from: <https://www.docker.com/> [Accessed: 13 August 2019]
- [4] Node-RED [Internet], 2019. Available from: <https://nodered.org/> [Accessed: 13 August 2019]
- [5] Highcharts [Internet], 2019. Available from: <https://highcharts.com/> [Accessed: 13 August 2019]
- [6] Correia FM, d'Angelo JV, Almeida GM, Mingoti SA. Predicting kappa number in a Kraft pulp continuous digester: A comparison of forecasting methods. *Brazilian Journal of Chemical Engineering*. 2018;**35**(3):1081-1094
- [7] Rahman M, Avelin A, Kyprianidis K, Jansson J, Dahlquist E. Model Based Control and Diagnostics Strategies for a Continuous Pulp Digester. In *PaperCon*, 2018
- [8] Pulp and paper (Billerudkorsnäs)—FUDIPO [Internet], 2019. Available from: <https://fudipo.eu/pulp-and-paper/> [Accessed: 12 August 2019]
- [9] Rahman M, Avelin A, Kyprianidis K, Dahlquist E. An approach for feedforward model predictive control for pulp and paper applications: Challenges and the way forward. In: *InPaperCon 2017*; 23-26 April 2017. Vol. 10. Minneapolis, Minnesota, USA: TAPPI Press; 2017. pp. 1441-1450
- [10] Hahn A. Operational technology and information technology in industrial control systems. In: *Cyber-Security of SCADA and Other Industrial Control Systems*. Springer Champions; 2016. pp. 51-68
- [11] Morris HD, Ellis S, Feblowitz J, Knickle K, Torchia M. A Software Platform for Operational Technology Innovation. *International Data Corporation.*; 2014. pp. 1-7
- [12] Balalaie A, Heydarnoori A, Jamshidi P. Microservices architecture enables devops: Migration to a cloud-native architecture. *IEEE Software*. 2016;**33**(3):42-52
- [13] Färber F, May N, Lehner W, Große P, Müller I, Rauhe H, et al. The SAP HANA database—An architecture overview. *IEEE Data Engineering Bulletin*. 2012;**35**(1):28-33
- [14] Aslanidou I, Zaccaria V, Rahman M, Oostveen M, Olsson T, Kyprianidis K. Towards an integrated approach for micro gas turbine fleet monitoring, control and diagnostics. In: *Global Power and Propulsion Forum 2018*; Zurich, Switzerland. 2018
- [15] Rahman M, Zaccaria V, Zhao X, Kyprianidis K. Diagnostics-oriented modelling of micro gas turbines for fleet monitoring and maintenance optimization. *PRO*. 2018;**6**(11):216
- [16] Azure—Microsoft [Internet], 2019. Available from: <https://azure.microsoft.com/en-us/> [Accessed: 13 August 2019]