Digital Twin-enabled Application Architecture for the Process Industry

6 authors, including:

Christoph Nölle
VDEh-Betriebsforschungsinstitut
2 PUBLICATIONS   **0** CITATIONS

SEE PROFILE

Asier Arteaga
Sidenor Aceros Especiales
17 PUBLICATIONS   **53** CITATIONS

SEE PROFILE

Antonio Salis
Engineering Ingegneria Informatica
22 PUBLICATIONS   **70** CITATIONS

SEE PROFILE

Gabriele De Luca
Engineering Ingegneria Informatica R&D
5 PUBLICATIONS   **34** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

H2020 mF2C View project

SMARTER - Steam and gas networks revamping for the steelworks of the future (RFCS - G.A. 101034060) View project

Digital Twin-enabled Application Architecture for the Process Industry

Christoph Nölle[1], Asier Arteaga[2], Joseba Egia[3][a], Antonio Salis[4][b], Gabriele De Luca[5]
and Norbert Holzknecht[1]

[1]*VDEh-Betriebsforschungsinstitut (BFI), Sohnstraße 69, 40237 Düsseldorf, Germany*
[2]*Sidenor I+D, Barrio Urgate, 48970 Basauri, Spain*
[3]*Mondragón Sistemas, S.COOP., Ama Kandida Etorbidea 21, 20140 Andoain, Spain*
[4]*Engineering Ingegneria Informatica S.p.A., Loc. Sa Illetta, SS195 km 2,3, I-09123 Cagliari, Italy*
[5]*Engineering Ingegneria Informatica S.p.A. Via Monteroni s.n., C/O Edificio Dhitech, Ecotekne, I-73100, Lecce, Italy*

Keywords: Digital Twin, Process Industry, Steel, Ontology, Cognitive Solution, Open Source.

Abstract: We develop a software platform architecture for the integration of heterogeneous software applications in the process industries, ranging from physical simulation models to data-driven AI applications and visualizations. Digital twins play a key role in this architecture, providing harmonised interfaces to a diverse set of data sources, based on a domain-specific data model rooted in standard ontologies. We investigate the applicability of existing standards and open-source software, demonstrating on the one hand their immense potential for software-based innovations in the process industries, but also highlighting some shortcomings and the need for further developments. Finally, a concrete implementation and data model for the production of steel long products is presented.

1 INTRODUCTION

Process industries nowadays collect large amounts of process data from sensors and machines, which can be exploited for various software-based innovations, in particular when combined with additional data sources, such as the enterprise resource planning system (ERP). Applications include among other advanced analytics, such as root cause analysis for defects and anomaly detection, process control, planning optimizations, such as predictive maintenance, and in general an improved observability of the processes based on visualizations and real time simulations (Mowbray et al., 2022).

In practice, several obstacles need to be overcome to make best use of the available data and allow for its efficient usage, such as different communication protocols and data formats used by different systems and a lack of semantic information or missing context information. Raw sensor data is usually time-based, but for many types of analyses an asset-centric view

of the data is needed, referring for instance to the product id and a position on the product surface, instead of time. Such conversions can be difficult, error-prone, or impossible to implement for application developers at all. For example, in the steel production, the tracking of individual products through the complete process chain is an important prerequisite for such a conversion.

The concept of Digital Twins encompasses, among other aspects, an asset-centric view on the data. They are meant to break up the existing information silos by providing a common interface to data originating from a diverse set of sources, provide a runtime for models related to the asset, and offer services to other twins, applications, and the user of the system (Jacoby et al., 2021). Digital Twins can thus serve as an important enabling technology for innovative software applications in the process industries. In this paper we outline the design of a modular software platform architecture that centers around Digital Twins.

---

[a] https://orcid.org/0000-0001-5851-8548
[b] https://orcid.org/0000-0002-4012-7490

The main goal of the platform is to enable the development of various kinds of applications, from data-driven to physical simulation models and from pure visualizations to machine controllers and optimizations. We emphasize that particular attention should be paid to the design of the application programming interfaces (APIs) offered by the system (Autiosalo et al., 2021). While on the one hand they should be kept as simple as possible to enable efficient app development, they need to fulfil several important additional requirements:

- provide access to different types of data, such as structured data, timeseries, or multimedia
- enable filtering of data by different categories, such as product ids and product properties, machine ids or time (geo-spatial properties could be considered as well, but are less relevant in our use case)
- provide semantic information and metadata, based on a well-defined domain-specific data model
- provide separate views of an asset related for instance to different life cycle phases, such as design, engineering, production, commissioning, operations, maintenance, service and end of life (Boss et al., 2020)
- allow for fine-grained access-control rules
- be easily extensible (in backwards-compatible ways, where possible) and modular
- be interoperable with existing tools, by leveraging on standards and best practices

Process industry companies planning to setup a central data access platform have several implementation options to consider. These include proprietary IoT platforms, offered for instance by the big cloud providers, open-source data brokers and streaming platforms, such as the tools from the Apache ecosystem (Kafka, Spark, Flink, StreamPipes, NIFI, etc) or more specific solutions with a more limited scope, such as an OPC UA server. In the final section of the paper, we present one concrete implementation based mostly on open-source components that has been developed in the CAPRI project for a steel production plant.

2 INTERFACES AND DATA MODEL

Since our software platform is meant to allow for a modular development of digital twins and to serve various kinds of applications, a particular emphasis

must be placed on the application programming interfaces (APIs).

2.1 Data Types

Different types of data arise in the context of a production plant, including

a) Structured data
b) Temporal data, i.e., data that changes in time, such as measurement values
c) Timeseries data, including historical data and forecasts
d) Multimedia data, including audio and video streams

Both classical timeseries and multimedia data should support an asset-centric interface (access via Digital Twins, possibly both machine twins, product twins and others), besides a simple time-based filter. For instance, out of a stream of images taken by a camera, a user or application might want to work with those images showing a particular product, and this filtering capability would ideally be implemented by a dedicated platform module and not by the consuming application. We will hence need advanced filtering possibilities; a simple file download capability would be insufficient.

Geo-spatial data is often added as a further category but is not universally required.

2.2 Protocols

Due to the ubiquity of HTTP-based REST services and the JSON data format on the internet-era software world the use of HTTP as a transport protocol and JSON as a serialization format, at least for structured data, should be undisputed. In addition, the JSON Linked Data (JSON-LD) may be used for providing context information.

For event-based access methods HTTP alone is not sufficient, however. Websockets and MQTT seem to be the major competitors in this field. Given that the websocket protocol is defined as an extension of HTTP it appears as the most obvious choice for an event-based API.

Another important aspect of an API is the query language. As mentioned before, the API must support complex filtering operations, which also depend on the type of data requested. Whereas developers from an ontological background may prefer the SPARQL query language, ease of use in a variety of applications seems to dictate the use of JSON-based mechanisms for queries as well. The most common approach for complex filters should be to define

queries as JSON objects in the HTTP body of a POST request.

Another interesting model for query languages that plays nicely with JSON is Facebook's GraphQL. It is particularly suitable for data represented in the form of graphs and allows the user to specify the data representation in a very flexible way, adapted to the needs of the application. Although it is not as common as REST interfaces, it has quite a significant user base and good tool support and is a promising technology for empowering data-driven applications on graph-structured data. GraphQL has been explored in the context of Digital Twin applications before, see for example (Autiosalo et al., 2021).

2.3 Standards

Different initiatives exist which develop standardized APIs for Digital Twins and related technologies, including the Asset Administration Shell (AAS) (Boss et al., 2020), (Plattform Industrie 4.0, 2021), the NGSI-LD standard by ETSI (ETSI, 2021), which has its roots in the FIWARE initiative (FIWARE Foundation, 2021), and the Web of Things (WoT) developed by W3C (W3C, 2020).

Proprietary platforms targeted at Industrial IoT and Digital Twin applications are, among other and in alphabetical order, Amazon AWS IoT, Bosch IoT Suite, Hitachi Lumada, Microsoft Azure IoT, PTC ThingWorx, Siemens MindSphere, and Software AG's Cumulocity IoT platform. In our case there is a requirement to base the platform on open-source components to avoid a lock-in effect and to enable market access to small and medium sized companies, which is why we will not further investigate these commercial platforms here. A comparative evaluation including them will be an interesting proposal for future work.

It should be noted that Microsoft has published a specification called Digital Twin Definition Language (DTDL) (Microsoft, 2019), which describes the modeling principles behind Azure IoT services and is somewhat similar in nature to the NGSI-LD model. Whereas the open publication of the specification is certainly useful and (Jacoby & Usländer, 2020) include it in their analysis, we decided not to include this proprietary model here.

On the one hand, the generic nature of standard interfaces can be useful from the point of view of interoperability. On the other hand, these interfaces are typically not domain-specific and not tailored to the situation at hand, possibly leading to inefficient and non-obvious query patterns, which is why domain-specific interfaces are generally considered a

good-practice in software development. In the case of NGSI-LD, the proposed way to enhance the interface with domain-specific information is to provide a separate domain model describing the possible properties and relationships that entities can have, in terms of JSON schema files. A similar approach can be taken for the AAS. This does not remedy all the problems with the generic nature of the interfaces, however. WoT takes a different approach in that it provides an interface description language which can be used to support multiple different APIs. The idea behind this approach is to integrate multiple devices which already come equipped with an API of their own. This setting appears to be less relevant to our scenario.

An analysis of different standards from the point of view of digital twin applications has been published in (Jacoby & Usländer, 2020). Among other criteria, they considered the support for different data types (geo-spatial, temporal, and timeseries, besides normal structured data), the possibility to describe custom events and services, and the query language.

The survey finds that NGSI-LD has good support for different data types (geo-spatial, temporal and timeseries) and a reasonable metamodel, but falls short of describing custom services and events, an aspect we consider of lower importance for our case. AAS on the other hand, lacks in support for data types, but covers services and events descriptions for submodels (Plattform Industrie 4.0, 2021).

The handling of multimedia data seems not to be covered by the existing standards, but to fully integrate with the asset-centric Digital Twin platform, advanced filtering possibilities will be needed, and it could be beneficial to standardise those as well (BDVA, 2017).

All the standards investigated in (Jacoby & Usländer, 2020) support JSON via HTTP, and the JSON-LD is also commonly used. For filtering, both NGSI-LD and AAS mainly rely on query parameters instead of request bodies, and AAS even defines several query parameters as BASE64-encoded, complex JSON objects. NGSI-LD on the other hand provides an alternative query interface that accepts POST requests with filters in the body. The query objects are only partially adapted to this setting, however, and may still require lengthy and error-prone string concatenations for complex queries.

A GraphQL interface is not foreseen by any of the specifications mentioned above, but since the NGSI-LD metamodel already assumes a graph-like structure, these two technologies might still be a good fit.

2.4 Semantics

A digital twin aims to provide an accurate representation of a physical or logical asset. The underlying data model of this representation will cover several technical and possibly business concepts at different levels of detail, depending on the context and aims of the platform. The data model should be clear and consistent, and modular in the sense that concepts relevant to a particular subdomain can be properly encapsulated. All submodels should be based on a common top-level model, however, providing generic concepts such identities, timestamps and locations (ETSI, 2019), and possibly also one or multiple standardised domain-specific base models (ETSI, 2019), (DKE, 2020). Generic top-level models could include the oneM2M base ontology (oneM2M, 2019) or the NGSI-LD cross domain ontology (ETSI, 2021).

Standardized domain-specific models for industrial applications include SAREF4INMA (ETSI, 2020), AutomationML[3], OPC UA[4], IEC 61360 Common Data Dictionary (CDD)[5], and ECLASS[6].

If automated reasoning or formal model validation is a requirement for the digital twin platform, then instead of a simple data model the use of ontologies and supporting technologies should be mandated. We will however not pursue this aspect here.

Tooling to support the usage of the data model in applications include machine-readable documentation, based for instance on JSON schema and OpenAPI, and clients or software development kits (SDKs) for specific programming languages. The OpenAPI approach is advocated for by FIWARE and the Smartdatamodels initiative (FIWARE Foundation, 2021). An alternative or supplementary model description can be realized in terms of a GraphQL schema, mandatory for developing a GraphQL interface, which emphasizes the possible relationships between entities.

3 PLATFORM ARCHITECTURE

An overview of the proposed software architecture is shown in Figure 1. At its core is the Digital Twin platform, which provides APIs to the twin applications; the actual twins of machines, products and possibly other assets are composed of these applications, while their state is stored in the twin persistence layer, labeled Twin DB in the figure.

Partly, this state may consist of references to the raw data, stored in the Raw DB layer.

3.1 Types of Applications

We consider three types of applications at different positions of the architecture. **Edge apps** run close to the actual IoT devices, for instance on an IoT gateway or on a device itself. Typical examples of this class are applications with low latency requirements, or those that process large amounts of raw data which shall not be forwarded as a whole to the upper platform components. An example from our steel use case could be an app that recognizes id tags attached to steel products from a camera stream. The id information and possibly some information about the recognition quality will be forwarded to the data broker and digital twin, but not necessarily the raw image data. Edge apps access data in the format it is available and do not require tailored APIs, which is why we are not going to consider them further in this work.

Broker apps operate at the next level, being connected to the data broker. They typically operate on the raw data before it is being persisted and do not use the Digital Twin APIs. Examples can include data quality validation tools, which attach metadata to the raw data stream. Broker apps may have higher processing requirements than edge apps and may need to access data from multiple sources.

Finally, we have the **Twin apps**, which provide the core of the Digital Twin functionality, making use of the APIs provided by the twin platform. We consider this the default case, where the application developer will not need specific knowledge about implementation details of the platform, instead use only well-documented interfaces.

The category of twin apps is very broad and could be further subdivided, for instance into visualizations, simulations, external data connectors, etc.

3.2 Data Interfaces

As discussed in Chapter 2 the twin platform offers a set of data-centric APIs to the twin applications:

1. *REST CRUD API*, or *Twin API*: the basic interface for retrieval, creation, and updates of digital twins.
2. *GraphQL API*: provides a graph-based view of the digital twins, particularly suited for representing complex relationships between

---

[3] https://www.automationml.org
[4] https://opcfoundation.org
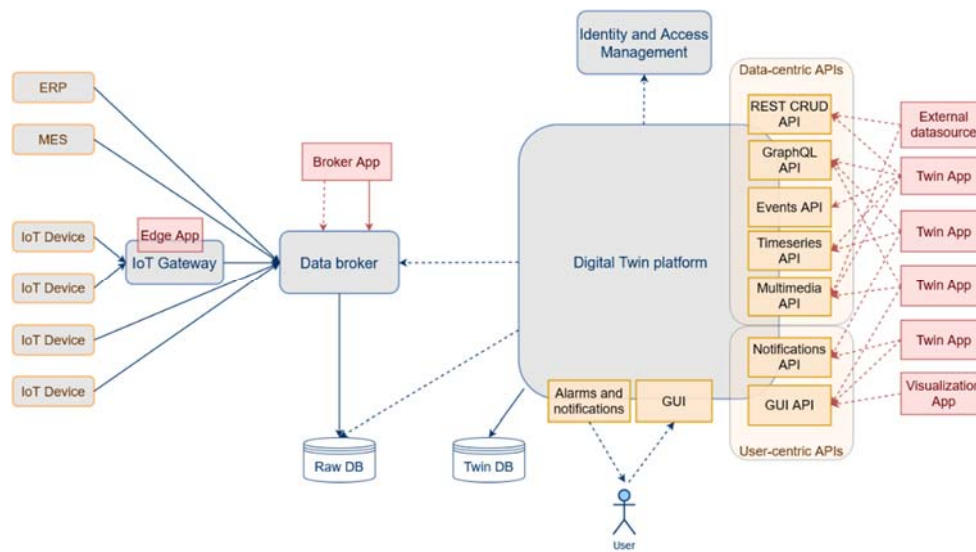[5] https://cdd.iec.ch
[6] https://eclass.eu/

Figure 1: Overview of the system architecture with a focus on the Digital Twin platform and its application and user interfaces. Data from the IoT devices flows into a central data broker, where it can be accessed by streaming applications, performing for instance a plausbility analysis. It is then stored in a dedicated storage layer labelled *Raw DB*, consisting of one or multiple databases. The Digital Twin platform accesses the raw data via these databases, and potentially also directly via the broker for real-time access (serving the Events API). Since the data flow from broker and raw db layer to the twin platform happens on-demand, it is visualized by means of dashed arrows originating from the platform, as opposed to the continuous stream of data from the devices to the broker shown as solid lines.

multiple assets. This may be realised as a read-only interface.

3. *Events API*: allows applications to register listeners on individual data points or classes of data points, enabling a push-based or reactive programming style. This is a read-only interface.

4. *Timeseries API*: receives queries for historical data, forecasts, etc., offering among others filtering and aggregation capabilities. Access to historical process data is typically read-only, whereas forecasts and other kinds of timeseries require write-access, too.

5. *Multimedia API*: allows for the retrieval of image, video, or sound data, offering filtering capabilities aligned with the twin structure. Depending on the setup, this may be realized as a read-only interface.

The Twin API (1) provides references to the timeseries API (4) and multimedia API (5), where appropriate, so that applications can find out how to access timeseries and multimedia data relevant to an asset, and it also needs to indicate which data points are eligible for access via the Events API (3). Documentation of the Twin API should include a data model description for the structure of the twins (pertaining to the REST API and GraphQL API).

Depending on the data sources available and the use cases to be realized with the platform, not all of the APIs need to be present in every implementation,

in particular the GraphQL API and the multimedia API can be considered optional.

3.3 User Interfaces

The user interaction with the twin platform typically centers around two concepts, a graphical user interface (GUI), actively controlled by the user, and an alarming facility that informs the user about unwanted or unexpected system states. Given the modular nature of our twin platform it is clear that the user interfaces need to be configurable and extensible by applications as well. The technology of choice for modern GUIs is the web platform, so the twin platform should provide a means of registering web applications, and likewise for registering alarm and/or notification targets.

Graphical user interfaces in the simplest case access the twin APIs for their data access. For example, an application that generates a forecast of the temperature evolution of a steel product on a cooling bed may store these forecasts in the twin platform via the timeseries API. The visualization provided by the same app would access this data via the timeseries API, as well. This mechanism will not always be sufficient, however, with a need for some applications to provide custom service interfaces to their web apps. In our example, the temperature forecast app could provide a means to simulate

259

certain "what if" scenarios, allowing the user to trigger a simulation with custom parameters and displaying the results on the web interface, without necessarily storing the results in the platform.

3.4 Interaction between Twin Applications

The modular nature of the twin platform, with functionalities provided by potentially many applications, raises the question of how these applications can interact with each other.

In the simplest case, different applications interact only via properties of the digital twins. For instance, App1 may set the value of a property *expectedLifetime* of a machine twin via the REST CRUD API, while App2 listens to changes of the same property via the Events API and reacts to the changes. This method will not cover every use case involving multiple independent applications, but where it is applicable it is an excellent approach in terms of interoperability and simplicity.

Where this is not sufficient, dedicated aggregation apps may serve the purpose to provide combined services, for instance running two simulations in sequence, where the results of the first simulation are used as input for the second. The drawback of this approach is that it only allows for static combinations of apps, which must all be known to the aggregator. One can also conceive of more a sophisticated platform which enables discoverability of custom services and standardizes the service interfaces, so that applications can in principle interact in a (semi-)autonomous way. For instance, one could think of an autonomous production planning service which interacts with the digital twins of different machines to find the optimum processing route for products, querying requirements and constraints from the order book and capabilities from the machine twins. While our architecture does not preclude such an approach, we do consider it a secondary priority for process industry scenarios. Similarly, it would be feasible to extend the platform to a co-simulation platform, enabling the user to combine and configure different simulations that would all need to adhere to a common specification, such as the Functional Mock-up Interface (FMI) (Modelica Association, 2022) for input and output data. This kind of advanced functionality should be kept out-of-scope for the basic platform and should rather be realized in terms of add-on modules, if needed.

3.5 Security

The applications foreseen for the presented concept providing information that belongs to a company's corporate secrets because deep insights to the production are given. Therefore, security is a main concern. As a first step, the platform and applications should only be accessible from within a companies' protected network, if possible, behind a firewall to the internet, so that only registered users can get access via an encrypted VPN tunnel. In addition, the zero-trust paradigm demands that all communication with and between services be protected.

Following the first law for web-programmers ("The user is not your friend") several techniques to improve the security are foreseen. Besides transport layer security (TLS) also authorization, authentication, and access controls need to be implemented. The principle of least privileges can ensure that not everyone has access to everything and performs only those actions they have authorization for. Depending on the company's security concept open-source tools for multi-factor authentication like Keycloak (Keycloak, 2022), PrivacyIDEA (privacyIDEA, 2022) or Google2FA (Google Two-Factor Authentication for PHP, 2022) can be used for increased authentication security avoiding 'lost' passwords due to social engineering.

Finally, already during the development the security must be considered using, e.g., application security best practices, such as the OWASP guidelines[7], patch management, etc.

4 OPEN-SOURCE IMPLEMENTATION

In the Horizon2020 project CAPRI (Cognitive Automation Platform for European PRocess Industry digital transformation) a reference architecture for cognitive-enabled automation platforms (CAPs) has been developed (Salis et al., 2022) along with three implementations for different process industry domains: asphalt, steel, and pharma. Here we report on how the digital twin concept has been integrated into the CAP for the steel domain.

Figure 2 above shows the data flow architecture proposed for the steel use case, with the following three main elements:
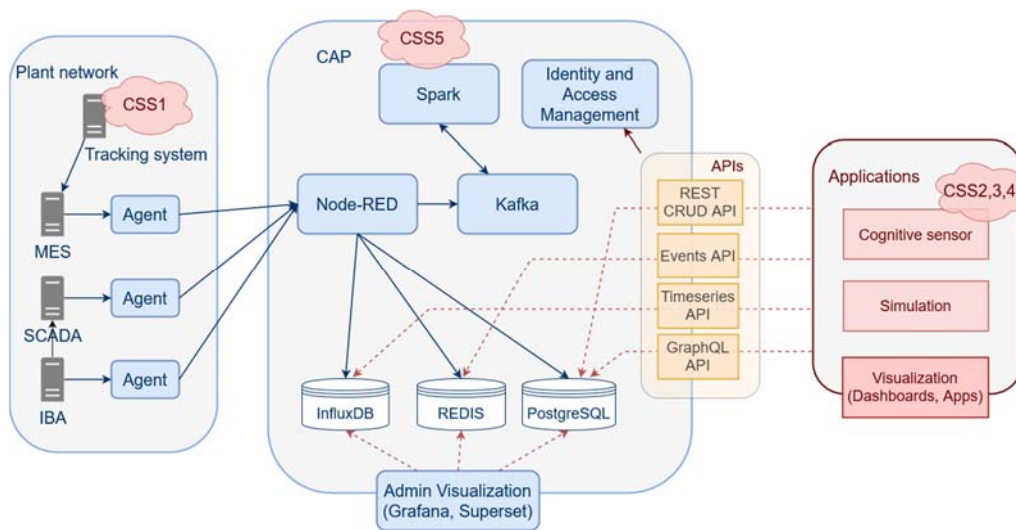
---

[7] https://owasp.org/

Figure 2: CAP implementation.

- the data capture through agents from the existing automation system
- the CAP, where the data ingestion, validation, storage, aggregation, and accessing is performed
- the Apps, which consume the raw and/or aggregated data from the CAP and return results to be consumed by other Apps

For the data capture the idea is to use Debian based Edge IIoT type equipment in which the main service is Node-RED. This free software based on Node.JS is a development tool based on flows for visual programming designed to facilitate the integration of hardware devices, APIs and online services as part of the Internet of Things. There is a very participative community that has developed and maintains libraries for communication with industrial devices and protocols of any kind, making it a very agile tool for development and testing but also allowing fast and easy replicability. In addition, the use of Node-RED has made it possible to recreate the data capture architecture in the form of a simulation of the process in a transparent way for the CAP, especially interesting in the development and validation stages of the apps.

The communication of the data captured on the Edge devices to the CAP has been implemented using MQTT TLS, the most widely used protocol for IIoT data transmission to which an end-to-end encryption layer has been added. In the CAP, once again Node-RED was selected to perform the functions of an ETL, as it allows to parallelize the work of ingesting the three types of data:

- Events and data in motion

- Timeseries
- Structured data

These three types of data are validated, transformed in Node-RED and finally stored in their corresponding databases respectively:

- Redis: in-memory data structure store, used as the key-value database of the data in motion.
- InfluxDB: a database for storing and retrieving time series data, very resource-efficient when dealing with large volumes of historical data.
- PostgreSQL: relational database management system that emphasizes extensibility and SQL compliance.

Finally, with the goal to allow different applications to access the data, a REST API has been developed and implemented, including endpoints for structured data (the twin API) and timeseries, as well as an events interface. The multimedia API proposed in Section 2.1 was foreseen initially, since the billet and bar tracking system installed in the plant generates images of the QR codes on tracked products that need to be analyzed. For the scope of this project, it was decided to perform the QR code recognition solely on dedicted edge computers installed at the shop floor, however, eliminating the need for any handling of images, videos, or sound files via the twin platform. It was hence decided that the multimedia API will not be provided in this project but might be added in a future expansion. The addition of a GraphQL interface is planned, too, it is already shown in Figure 2.

This service requires authentication by a token previously generated in the server, in which the corresponding read, write or r/w permissions are

261

assigned for each type of data and database table and token.

The necessary classes have been developed in such a way that the data consumers can obtain, in the three existing data types, the structures of the tables. Then, the classes for reading and writing these data types have been developed, each of them with their corresponding query structure requirements.

In addition to all this, the integration with the Kafka broker has also been implemented to make the CSS5 developments able to receive the process data in real time.

Finally, to complete the CAP architecture, several process monitoring visualizations have been created using Grafana (for time series) and Apache Superset (for aggregated and structured data). Both data visualization platforms are open-source and are gradually becoming standard services in the industry.

4.1 Data Model

The data model in our approach describes the structure of entities accessible via the REST API. It is specified in terms of JSON schema files, as advocated by the Smartdatamodels initiative and the FIWARE community (https://smartdatamodels.org/). The JSON schema files can be used to generate both OpenAPI/Swagger-compatible interface descriptions and a JSON-LD context file, which can then be served from the API web server. This way, interoperability with a variety of existing software tools and compatibility with the semantic web principles is ensured.

As the base ontology for our model, we selected the NGSI-LD cross domain ontology (ETSI, 2021), and the domain-specific model is based on SAREF4INMA, the SAREF for industry and manufacturing ontology (ETSI, 2020), with some adaptations to meet the needs of our specific use case. Figure 3 shows the high-level view of the model. One central class in the model is ProductionResource, which encompasses all the resources and (semi-)products used and produced in the process, such as the batches of hot, liquid steel transported in a ladle (class LadleHeat) and steel billets and bars (classes Billet and Bar). The sub-hierarchy of ProductionResources is shown in Figure 4. Another important class is ProductionEquipment, which models machines and plants. ProductionResources keep a reference to the equipment they have been treated with, but not vice versa. These references may also contain additional properties and are themselves submodels of the Transformation class.
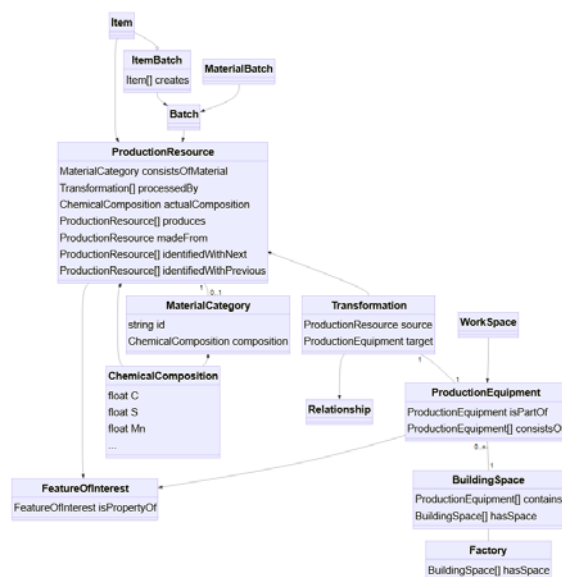

Figure 3: High-level data model for the steel production, loosely based on SAREF4INMA (ETSI, 2020).

One specific feature of our use case is the tracking uncertainty for steel billets and bars. After billets are cut from the casting strand of the continuous casting machine, they are usually stored for several days in a large storage facility before being further processed in the hot rolling mill. Similarly, steel bars created from the rolled billets in the hot rolling mill first go to a cooling bed and storage before being transported to the finishing line. A tracking system has been installed in the plant for identifying the individual billet, resp. bar, when entering the hot rolling mill, resp. finishing line, but due to the harsh conditions within the steel mill the tracking system cannot identify every item correctly, and a significant error rate remains. In the model we handle this problem by maintaining separate digital twins for billets from the casting machine and those in the hot rolling mill, and by means of references between the two, dubbed *identifiedWithNext* and *identifiedWithPrevious*. These references can be multi-valued to deal with the possibility of tracking errors, although there should in principle be a 1-1 correspondence between the two.

Since our model allows for references of different kinds, it can be represented as a graph. Furthermore, since both entities and relationships, i.e., nodes and edges of the graph, can have properties, it is a property graph. An example is shown in Figure 5. The natural representation of the model in terms of a graph imply that a GraphQL-based API is very suitable for the development of applications.

It is therefore planned to add such an interface to the existing REST, timeseries and events interfaces.
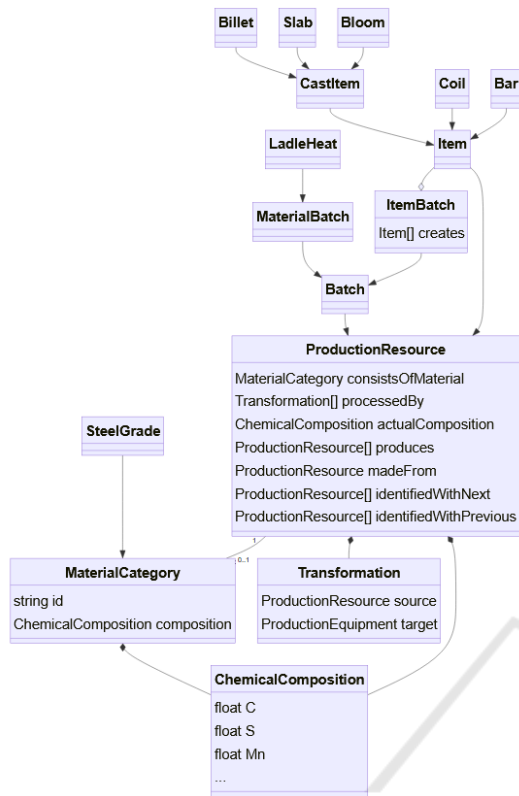
262

Figure 4: The ProductionResource hierarchy, showing some steel (semi-)products, such as Billets and Bars, at the top.

4.2 Applications

Being competitive in the worldwide steel sector is of great importance for the European steel companies. One solution for a differentiation is the production of high-quality products. For that a detailed knowledge of the actual state of each individual (semi-)product from the very beginning combined with an early detection of a risk for quality deficiencies will lead to an advantage. To reach this aim the following applications are being developed in the project:

- Product tracking (CSS1): the tracking of the products along the production chain considering the changes in the product geometry. This is mainly realized as an edge application, i.e., the relevant software is installed near the marking and reading hardware.
- Data enrichment: soft sensors for steel solidification in the continuous casting machine (CSS2), internal temperature development in the hot rolling mill (CSS3) and scale development (CSS4). These are realized as twin applications,

i.e., they use the APIs of the CAP for data access.
- Risk sensor (CSS5): online evaluation of the available information to early detect a risk for degraded product surface quality. This is realized as a broker app, i.e., it operates directly on the incoming data streams within the CAP.

The tracking system in the steel production has been improved within the scope of this CAPRI project, and this is a helpful tool for the Digital Twin architecture application, as the steel goes through severe transformations during the global process from liquid steel to final bar, with an intermediate step of billet, in this case. The initial state was a tracking to the heat level: every billet and bar had a clear identification tracked up to the liquid steel moment. This way, the chemical composition and most critical parameters are tracked and ensured. Nonetheless, there are some other parameters that depend on process data that vary within one heat as they are produced in the solid phase, so different bars from the same heat number can have different casting speeds or reheating processes.

A more detailed tracking was designed to be able to identify in each of the final bars the significant parameters happening in the different moments of the process.

The concept of this tracking is based on a combination of hardware and software tools to produce a bridge of the material in each of the transformations and a clear identification after it. The main difficulties arise from the temperature conditions and, in some cases, the speed of the process. The approach followed is to mark every billet in a hot state after it is cut at the end of the Casting Machine, engraving a QR code by means of a laser (see Figure 6). In the next step this billet, already identified, is read in the entrance of the Rolling Mill, goes through the rolling transformation into a bar and each bar is marked again, with billet and bar identification, in hot conditions. Finally, the bar, in cold conditions, is read in the finishing units that check quality bar by bar. The last part of the system was successfully tried but not fully implemented as the Rolling Mill was revamped and the new conditions required an important adaptation of the marking system.

Apart from the hardware itself, the tracking requires important software considerations.

First of all, the Level 2 systems involved have to transmit the relevant new information and communicate with the markers. Reading cameras on
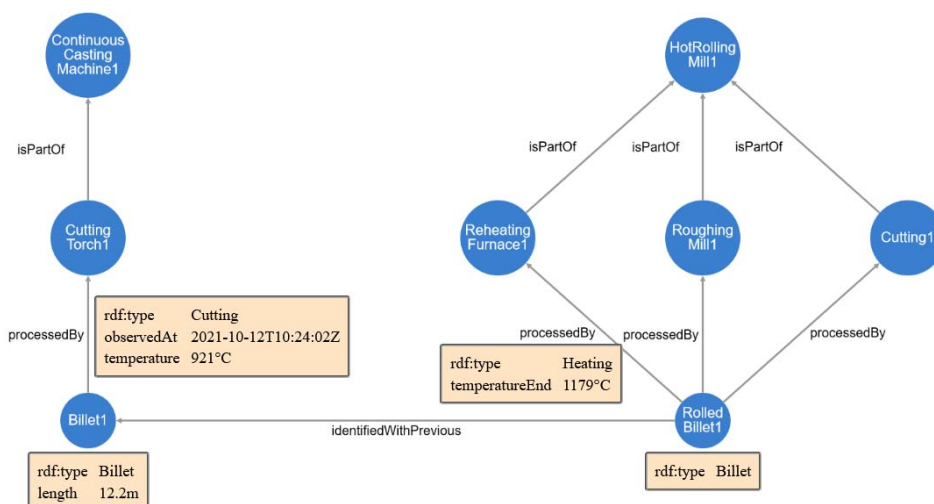
Figure 5: Excerpt of a data graph for the CAPRI steel use case.

Figure 6: A marked steel billet.

the other hand feed the Level 2 systems with the information obtained from the reading of the identified QR codes. However, there is another important aspect that has been addressed: the transformation from the time series data to the product-based data. Effectively, in the Casting Machine, for example, there are several billets in the different parts of the machine and at the same time one billet is in the mould, another one in the secondary cooling and another one, maybe from the previous heat, in the straightening unit. The transformation from the time series data to the billet-based data, has to be done strand by strand; it is important to correctly feed the Digital Twin; and it is a direct consequence of the tracking improvement, as previously, the only possible consideration was the average value for the whole heat.

The Digital Twins concept is implemented in our use case for the steel billets and bars, and it is heavily rooted in the improved tracking system. An example of billet and bar twins with an indication of their tracking status (successful, missed, duplicate) is shown in Figure 7.

The realization of broker apps, notably the risk sensor CSS5, in the platform is based on the coupling of Kafka data streams and Spark jobs running the applications. Kafka data streams comprise different data streams, including metallurgy, casting and hot rolling data, which arrive at different frequencies and volumes. Kafka jobs requests are structured in different topics.

The broker applications are organized as a set of spark jobs in three different steps, see also Figure 8:

- **Pre-processing**: this runs on the spark master and is in charge of collecting data coming from different sources (metallurgy and casting data), synchronizing data streams, loading to a HDFS and coded with a heatnumber. Algorithm correctness is ensured by the correct association of metallurgy and billet data in the correct sequence.

- Once both casting and metallurgy data are available the processing step can start.

- **Processing**: this part is in charge of receiving Kafka jobs. Each time the topics contain a new job, the spark job consumes it by accessing HDFS for data, and once all data is available, it fires the cognitive solution processing on different worker machines in parallel, leveraging on the CAP capabilities.

- **Post-processing**: once the output is generated to the spark worker node, the data is stored in the CAP persistence layer and sent back to the requester through a Kafka topic. The data stored will be later used to visualize all generated data. Custom dashboards can be designed and implemented allowing users to consume real-time and historical data for monitoring and decision support. Furthermore, a bidirectional interaction is supported,

Figure 7: Visualization of billet and bar twins in the steel use case. Rectangles in the upper row represent billets (steel semi-products) and those in the middle row represent bars (the final products) made from a single billet. Colors indicate the tracking status (green: successful, grey: missed, orange: duplicate).

enabling the control functionalities through the communication with the brokering layer as an intermediary for the southbound devices.

5 CONCLUSIONS

We have outlined the software architecture for a modular Digital Twin application platform for the process industries, along with an implementation in terms of open-source components that has been created for a producer of steel long products. We emphasized the need to support different types of applications and to provide well-documented and simple to use interfaces, enriched with semantic information and cross-referencing each other.
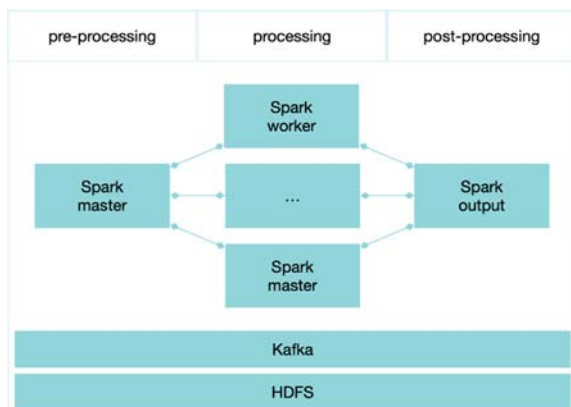

Figure 8: Detail view of the broker application runtime.

An analysis of the existing standards landscape for digital twins reveals the existence of promising initiatives, such as the Asset Administration Shell (AAS) or the NGSI-LD specification, but we also identified gaps and a few questionable design decisions in these rather new technologies, which ultimately drove our decision for a custom interface specification. Future iterations of these standards may improve on this and make them a good basis for achieving interoperability between systems.

The importance of product tracking for the application of Digital Twins concepts in the process industries cannot be overestimated. A novel tracking system for steel long products has been deployed in the steel plant considered, and our twin platform has been equipped with a data model that is well capable of representing tracking errors and uncertainties.

As a next step the platform and applications will be fitted for online operation and be deployed to the plant network. The goal is to significantly reduce the amount of surface defects, resp. to detect defective products early on, in order to avoid the costs incurred by the further processing and to reduce the impact on the environment. A set of KPIs has been defined to quantify the impact of these solutions.

Surface defects on the bars, as final products considered here, can originate from different processes in the steel mill, in particular from inclusions that occurred during the casting of steel or from problems in the hot rolling mill. The digital twin concept based on the new tracking system will allow us to retrieve the relevant datasets applicable to an individual bar, to analyse them for irregularities and potentially to identify the root cause of the problem.

ACKNOWLEDGEMENTS

REFERENCES

Autiosalo, J., Ala-Laurinaho, R., Mittal, J., Valtonen, M., Peltoranta, V., & Tammi, K. (2021). Towards Integrated Digital Twins for Industrial Products: Case Study on an Overhead Crane. In: *Applied Sciences*, 11(2), 683. https://doi.org/10.3390/app11020683

Big Data Value Association (BDVA) (2017). Strategic Research and Innovation Agenda. http://bdva.eu/ sites/default/files/BDVA_SRIA_v4_Ed1.1.pdf

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

DKE (2020). Industry 4.0 Standardization Roadmap. https://www.dke.de/en/areas-of-work/ industry/standardization-roadmap-industry-40

Microsoft (2019). Digital Twin Definition Language https://github.com/Azure/opendigitaltwins-dtdl

ETSI (2019). ETSI TR 103 535 SmartM2M; Guidelines for using semantic interoperability in the industry. https://www.etsi.org/deliver/etsi_tr/103500_103599/1 03535/01.01.01_60/tr_103535v010101p.pdf

ETSI (2020). ETSI TS 103 410-5 SmartM2M; Extension to SAREF; Part 5: Industry and Manufacturing Domains. https://saref.etsi.org/saref4inma

ETSI (2021). ETSI GS CIM 009 Context Information Management (CIM); NGSI-LD API. https:// www.etsi.org/deliver/etsi_gs/CIM/001_099/009/01.04. 02_60/gs_CIM009v010402p.pdf

FIWARE Foundation (2021). FIWARE for Digital Twins. https://www.fiware.org/wp-content/uploads/ FF_PositionPaper_FIWARE4DigitalTwins.pdf

Google Two-Factor Authentication for PHP [Computer software]. (2022). https://github.com/antonioribeiro/ google2fa

Industrial Internet Consortium (2020). Digital Twins for Industrial Applications. Definition, Business Values, Design Aspects, Standards and Use Cases. https://www.iiconsortium.org/pdf/IIC_Digital_Twins_ Industrial_Apps_White_Paper_2020-02-18.pdf

Jacoby, M., & Usländer, T. (2020). Digital Twin and Internet of Things - Current Standards Landscape. In: *Applied Sciences*, 10(18), 6519. https://doi.org/ 10.3390/app10186519

Jacoby, M., Jovicic, B., Stojanovic, L., & Stojanovic, N. (2021). An Approach for Realizing Hybrid Digital Twins Using Asset Administration Shells and Apache StreamPipes. In: *Information*, 12(6), 217. https://doi.org/10.3390/info12060217

Keycloak (Open Source Identity and Access Management) [Computer software]. (2022). https://www.keycloak.org/

Mowbray, M., Vallerio, M., Perez-Galvan, C., Zhang, D., Del Rio Chanona, A., Navarro-Brull, F.J. (2022), Industrial data science – a review of machine learning applications for chemical and process industries. In: *React. Chem. Eng.*, 7, 1471-1509. https://doi.org/ 10.1039/D1RE00541C

Modelica Association (2022). Functional Mock-up Interface (FMI). https://fmi-standard.org/

oneM2M (2019). Base Ontology. https://www. onem2m.org/images/pdf/TS-0012-Base_Ontology-V3_7_3.pdf

Plattform Industrie 4.0 (2021). Details of the Asset Administration Shell - Part 2. Interoperability at Runtime – Exchanging Information via Application Programming Interfaces. https://www.plattform-i40.de/IP/Redaktion/EN/Downloads/Publikation/Detai ls_of_the_Asset_Administration_Shell_Part2_V1.html

privacyIDEA (Flexible, Open Source Multi Factor Authentication (2FA)) [Computer software] (2022). https://www.privacyidea.org/

Salis, A. Marguglio, A., De Luca, G., Gusmeroli, S. and Razzetti, S. (2022). An Edge-Cloud based Reference Architecture to support cognitive solutions in the Process Industry. https://arxiv.org/abs/2202.06622

W3C (2020). Web of Things (WoT) Thing Description. https://www.w3.org/TR/2020/REC-wot-thing-description-202004.