OPEN MANUFACTURING
PLATFORM

Semantic Data Structuring

A publication by the Open Manufacturing Platform

Published
2021-08-16

www.open-manufacturing.org

info@open-manufacturing.org

Legal Disclaimers

"THESE MATERIALS ARE PROVIDED "AS IS." The parties expressly disclaim any warranties (express, implied, or otherwise), including implied warranties of merchantability, non-infringement, fitness for a particular purpose, or title, related to the materials. The entire risk as to implementing or otherwise using the materials is assumed by the implementer and user. IN NO EVENT WILL THE PARTIES BE LIABLE TO ANY OTHER PARTY FOR LOST PROFITS OR ANY FORM OF INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER FROM ANY CAUSES OF ACTION OF ANY KIND WITH RESPECT TO THIS DELIVERABLE OR ITS GOVERNING AGREEMENT, WHETHER BASED ON BREACH OF CONTRACT, TORT (INCLUDING NEGLIGENCE), OR OTHERWISE, AND WHETHER OR NOT THE OTHER MEMBER HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

ACKNOWLEDGMENTS
This document is a work product of the Open Manufacturing Platform – Semantic Data Structuring Workgroup, chaired by Birgit Boss (Bosch).

AUTHORS
Niklas Allard (Tetra Pak)
Soufiane Ameziane (Teradata)
Birgit Boss (Bosch)
Eric Charran (Microsoft)
Jean-Baptiste Courmont (capgemini)
Michal Dlubala (Cognizant)
Piotr Janik (Intel)
Mark Jaspan (AB InBev)
Eitan Kovacs (NI)
Pavan Kumar (AB InBev)
Sven Jeroschewski (Bosch)
Denis Molin (Teradata)
Lukas Römer (Bosch)
Andreas Rosengren (Cognizant)
Steffen Stadtmüller (Bosch)
Andreas Textor (Bosch)
Jürgen Vischer (Forcam)

CONTRIBUTORS
Manoj G L (WiPro)
Palanivel Jayakumar (WiPro)
Fabio Massari (VMware)
Torsten Mollien (P und Z)
Valerio Pisa (Reply)
Marco Rius Franscisco (Faurecia)

Contents

1 Scope

Semantic Data Structuring includes tools and methodology that organizations with a high level of technical and business maturity can implement. For these organizations, investment in such solutions will reduce integration costs, enable rapid decision-making in a changing production environment, and allow them to outsell the competition on the international market.

Quick access to information enables decision makers to make decisions on time, consolidating all the information available in the organization. The high quality of the information exposed by robust Semantic Data Structuring builds trust and confidence in decision-making at the management level of each organization. In this paper, the focus is on telemetry data and analytic problem solving. However, engineering data and product type data can also be modeled using the method presented in this paper.

Semantic Data Structuring can be used on any way that requires fast, orderly, and analyzed (Yes or No choice) material in the form of crucial information. Such a tool in the hands of decision makers reduces the probability of making bad decisions where time and information are number one.

2 Problem Symptoms

Quick access to databases and information from databases, in an orderly manner or utilizing mathematical algorithms (SDMs), facilitates quick analysis of available information, review and analysis of scenarios, testing, and modeling of mock-ups. Through decisions made by systems and staff, this information access allows the organization to quickly react to changing market conditions (demand, supply of goods, e.g., everyday use goods as well as luxury goods).

2.1.1 Challenges

Organizations that manufacture physical goods face several challenges when dealing with semantic data. These challenges can take many forms for both large and small companies:

- Many of these challenges are associated with understanding and increasing efficiency and optimization during production based on real-time data.
- Other challenges require focus on preventing harmful disruptions to the production process based on asset issues or unexpected maintenance that can represent disruptive and costly downtime. If contextualized data had been available, these unforeseen issues could have been avoided.

- In addition to avoiding production downtime, operational process challenges must be optimized by organizations so as not to experience inefficiencies during the manufacturing process, which can negatively affect production quality, rate, and cost-effectiveness.
- Additional challenges include how organizations respond to competitive pressure, search for new data-driven models for revenue generation and drive evolution and transformation.
- Finally, problems at a workplace become visible via data. These problems are often caused by supporting processes within and outside the company. A challenge is mapping the data to the responsible entities and changing/transforming those processes for the better.

Asset Diversity is very large and distributed.[1] Organizations that lack integrated data will be challenged to make decisions with only partial data or in an untimely manner. Many organizations face these challenges simultaneously which makes it difficult to determine where to invest to resolve them and in what order. It can be difficult to deal with the symptoms of these issues tactically and at the same time develop a solution to the underlying root cause of the issue. To completely bring the organization to a place of productive transformation it is important to get a holistic view of the challenges first. This should be done prior to engaging in complex, costly technical investments, which may result in temporary relief, but ultimately, end up being a brittle, fragile, and costly solution in the long run. The pyramid diagram 1-1 at the end of chapter 2.1.2 outlines the confluence of challenges faced by many manufacturers, while identifying clear requirements to address the challenges.

Organizations have a need to make data-driven decisions that can help them manage risk, optimize production, and adopt and explore new revenue streams or business models.

Data analytics that transform a business or organization, in which customers can explore metrics such as Overall Equipment Effectiveness (OEE), anomaly detection, and predictive maintenance, places organizations on the road to achieving significant ROI and marketplace competitiveness. Additionally, these analytics can be operationalized to the point where organizations can trigger optimization processes, AI training loops, and other processes to help increase efficiency and address costs.

2.1.2 Integration Investments and Effectiveness

To achieve transformation, information from CRM, ERP, and operational technology constructs within production facilities (inclusive of asset and process telemetry) are required. This combination of IT and OT data sources and the integration between them are foundational to achieve these analytical capabilities. However, many manufactures today have historically entrenched processes that are often backed by regulatory, safety and operational considerations and are not easy to change. Some

[1] Introduction to the OMP Manufacturing Reference Architecture General Approach & Design Considerations https://open-manufacturing.org/wp-content/uploads/sites/101/2021/05/OMP_Reference_Architecture_Whitepaper-17-May-21.pdf

organizations achieve a minimal amount of integration between the layers in the pyramid diagram below based on manual processes or manual integration. From humans with clipboards, Excel files, or other human-based processes, this integration is limited, although consistent and well understood. However, achieving data integration between IT and OT data sources is unattainable. This can be attributed to causes such as the manual nature of production processes and the isolated data it creates as a result.

Other organizations may have focused on a technology investment to help automate and integrate these layers. Significant investments like subscribing to vendor-supplied solutions create areas of capability but are often resistant to change when manufacturing or business processes change. These systems are highly schema-dependent, meaning that when the data format of any of these elements is updated or altered, the entire stack of technical integration must change as well, representing a significant cost.

To understand the challenge space the list below contains a set of commonly encountered problems by manufacturers.

- An inability to bring together data sets between the various systems of the manufacturing plant for capture or analysis
- Integration steps based on documented standard operating procedures that rely on excessive human-driven activities to enable the production process, spot challenges, or maintain the system
- A hard separation between IT data such as ERP, CRM, or SCM systems and production processes, thereby making an end-to-end analysis hardly achievable
- Limited understanding of OT (control systems) where these are a black box at delivery and developed by third-party vendor and data provided is not understood
- Silos of information that are rigid, locked to a specific instance of a historical process, and an inability to easily combine data for analytical impact
- An inability to understand equipment or facility effectiveness due to vendor or organizational data silos, separated by physical or process boundaries
- The amount of time required for data and analytics to be assembled, presented, and utilized by analysts is significant and complex
- Manufacturers do not dare to analyze data with a complex topology and big volume, e.g., sensor data may be very large in volume with low information density and requires complex processing to extract relevant information. e.g., press shops

Manual data collection and/or harmonization of data from different sources with different semantics prevent a holistic view of the factory. This missing transparency leads to ineffective activities and decisions.

- An inability to contextualize the data coming from plants so that it can be used in combination with IT data sources in an easily discoverable manner
- The existence of dark data (data that is gathered and stored but not used)
- Inability to give the insights on time because of too large data volume (large history) and complex processing (e.g., vibration use cases)
- Inability to join historical and real-time data inducing friction in AI model deployments trained on historical data and run on real-time data on the shop floor
- Inability to trust insights based on partial data (e.g., uncontextualized alert on the production line, the decision is the tradeoff between cost, delay, customer satisfaction, and scrap. All this context is coming from different sources and data domains)
- Difficult for the machine operator to find the manual and other documentation in case of failure or maintenance. Every machine has different failure codes
- Machine operator knows which data he or she wants to share with other parties. However, the data is not understandable to the other party without detailed documentation
- The component supplier wants data from the machine builder. However, the machine builder does not understand which data exactly would be transferred because the format is vendor-specific and not understandable

Machine builder wants data from the component supplier. However, the mapping to the data offered by the component and the language of the machine builder is very difficult and needs to be done for every single machine builder.

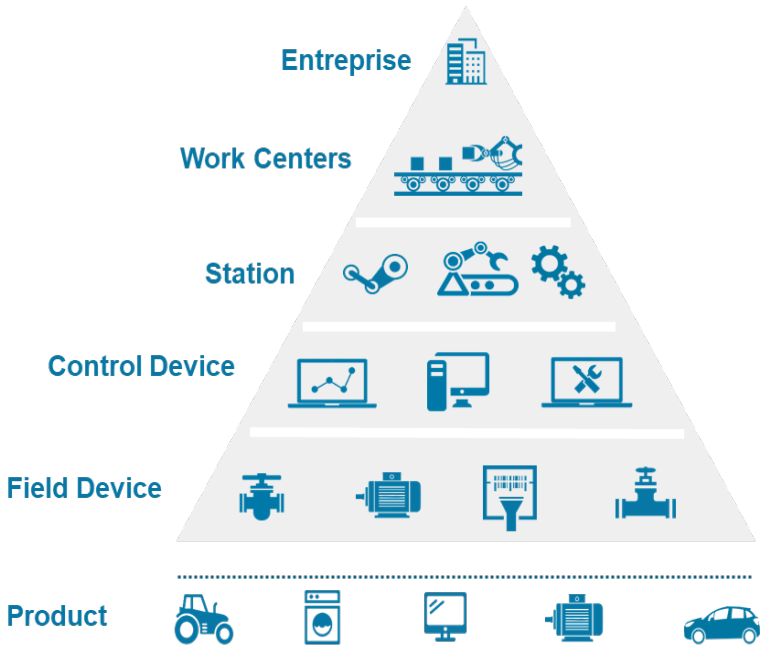

Figure 2-1 Only exemplary: Manufacturing Pyramid Industry 3.0 © Plattform Industrie 4.0

2.1.3 Automated Guided Vehicle Use case

A simple use case of a battery-powered Automated Guided Vehicle (AGV) will illustrate shopfloor level challenges. The AGV moves products from one place to another within the shop floor. To achieve this, the AGV follows a trajectory from source to destination, avoiding obstacles and other AGVs. The energy source is a battery that has a battery level (amount of charge remaining) and a warning level.

Figure 2-2 Automated Guided Vehicle Use Case © Bosch

Daily manufacturing production can be maximized by optimal AGV usage.
The challenges are as follows:

- Battery level
- Charging station proximity
- Charging station availability
- Required recharge time

One solution could be to schedule the battery charging plan, which is easy to manage but does not prevent battery power shortage or battery lifecycle optimization.

The idea is to model a digital twin of the battery level, which is going to manage the actual data of the battery with regards to its theoretical properties. Based on this model, the objective is to optimize maintenance activities of the battery or even predict battery behaviors.

3 Hypotheses for problem solving

Many of the problem symptoms listed in the chapter 2 have their causes 1) in the lack of understanding of the data and 2) in the lack of standardized formats between layers. In other words, what is missing is 1) to add understandable semantics to the data, to make information out of raw data, and 2) to provide standards for information exchange.

How can this be achieved? By explicitly defining the semantics in a model that is the same for the data. This model – or to be more precise, this meta model – should be standardized so that semantic interoperability can also be achieved when several partners are involved. The semantic definition of data can be identical even when the data itself or the format in which the data is submitted is different.

A known standard for defining the semantics of properties of product classes is IEC 61360. It is the basis for international data dictionaries like ECLASS[2] and IEC CDD[3]. These dictionaries had the focus on static and descriptive product data (example: manufacturer name or maximum rotation speed), but more and more properties for runtime data (example: measured rotation speed) have recently been defined. These dictionaries are a good input for reusing semantics defined by domain experts. However, even this will not solve all problematic symptoms, because it is still unclear which data decision makers or applications are getting in a specific point in time or context. So, what is needed is a meta model that allows the specification of the concrete content (context and time) for a specific asset.

Experience shows that it is neither flexible nor realistic to offer exactly one model for a specific product or asset with the aim of providing a complete and in-depth model. There will always be something missing, or the model is overloaded with information not needed. It is much more flexible and fulfills the real needs if the different aspects are defined or selected use case by use case. The application defines what data is of interest. Different applications may be interested in the same aspects, some might be specific for a special application. If needs change, additional aspects can be added at any point in time.

The Plattform Industrie 4.0 and the Industrial Digital Twin Association are starting to define so-called submodel templates[4] to enable interoperability via the Asset Administration Shell[5]. These are templates for different data and behavioral aspects a component or an asset can offer. This is exactly what is needed. A submodel template defines pairs of structural elements and a unique reference to its semantic definition. Such a semantic definition can be found – as discussed before – in data dictionaries like ECLASS and IEC CDD. The submodel templates already offer the required flexibility and

---

[2] https://www.eclass.eu/

[3] https://cdd.iec.ch/cdd/iec61360/iec61360.nsf

[4] Examples: “Submodel Templates of the Asset Administration Shell. Generic Frame for Technical Data for Industrial Equipment in Manufacturing”. V1.1. Federal Ministry for Economic Affairs and Energy (BMWi). URL: https://www.plattform-i40.de/PI40/Redaktion/EN/Downloads/Publikation/Submodel_templates-Asset_Administration_Shell-Technical_Data.html and “Submodel Templates of the Asset Administration Shell. “ZVEI Digital Nameplate for industrial equipment”. Version 1.0. Federal Ministry for Economic Affairs and Energy (BMWi). URL: https://www.plattform-i40.de/PI40/Redaktion/DE/Downloads/Publikation/Submodel_Templates-Asset_Administration_Shell-digital_nameplate.html

5 Asset Administration Shell Specifications. Federal Ministry for Economic Affairs and Energy (BMWi). URL: https://www.plattform-i40.de/PI40/Redaktion/EN/Standardartikel/specification-administrationshell.html

exactly describe which information the application will get. They have limits when there are no properties or models defined in a publicly available dictionary. For this reason, the Asset Administration Shell offers model internal concept descriptions but so far only properties conformant to IEC 61360 are supported.

We did not yet mention it explicitly, but we talked about products and assets. So implicitly, we already assumed that applications are not interested in "everything" but typically they are interested in data that is available from a specific device type in a factory (but typically abstracts from its manufacturer). Devices are assets but besides such physical entities also non-physical assets and more complex assets might be of interest to an application. For example, a complex machine or a complete factory might be an asset that an application is interested in. Or an application is interested in software artifacts or knowledge about humans with a specific role.

Here, the concept of a digital twin comes into place. A digital twin is a digital representation of an asset. Assets may be composed out of other assets. This also needs to be reflected in its and their digital twins. So, this is a very flexible and modular concept for providing data with regard to a specific asset. The twin additionally provides access control. With this, not all data is available to every potential user. As described before, a digital twin is a well-defined contact point to get information about the different aspects of the asset.

The big advantage of digital twins is their scalability. If the digital twin is standardized, the twin can be directly delivered together with a product of a supplier and just integrated with plug and produce into the environment of the OEM. This way enables that thousands of twins can be integrated that offer the same information although the devices/assets represented are very different and heterogeneous. The other way of scalability is the possibility to add additional aspects to the digital twin during its life span.

To be able to use a digital twin in the context of multiple systems and even cross-company borders, a strong identification concept is needed. Each party in an ecosystem of digital twins needs to understand how to find or address the right digital twin for its needs.

The hypotheses for solving the problematic symptoms are the following:

Hypothesis 1:

Data needs to be transformed into information.

Hypothesis 2:

Digital twins offer a modular and flexible approach to homogenize data of heterogeneity of different data sources (sensors, database, etc.). Modularity also supports the composition of digital twins for complex assets (production line, a machine composed out of components, robots, etc.)

Hypothesis 3:

A digital twin is a digital representation of an asset offering exactly the information and the models that are needed by the applications using the digital twins.

Hypothesis 4:

A flexible and scalable approach allows addressing new use cases during the life cycle of a product/asset. A digital twin is very flexible and can be extended if needed.

Hypothesis 5:

Standardization of digital twins as well as the metamodel for defining the semantics of data will further reduce integration costs.

Hypothesis 6:

A digital twin is an important touchpoint for an application to access information from various distributed sources about an asset based on a unique identifier.

In order to be able to express what kind of information a digital twin provides, a suitable modeling language or formalism is needed. This formalism must be usable to express both schema information and domain semantics. Schema information is used to validate data and includes statements about how the data must be structured, or which upper and lower numerical limits apply to a value. Domain semantics include statements that would otherwise only exist implicitly or informally, for example, which physical unit is used to measure a certain value.

Languages and corresponding technologies for expressing semantic models that can help fulfill these requirements and that are based on open specifications can be found in the scope of the "Semantic Web" initiative by the W3C. In particular, the "Resource Description Framework" (RDF) is a formalism intended for the specification of interlinked information models and data. As such, it is the prime candidate to be used as the basis for building the modeling language that is needed for digital twin data semantics[6]. Other open W3C specifications related to RDF can be used in the implementation: The SPARQL Protocol and RDF Query Language (SPARQL) can be used to query and transform RDF information models, and the Shapes Constraint Language (SHACL) can be used to specify validation rules for RDF information models. These W3C technologies are broadly adopted as de-facto standards for semantic modeling.

A DIGITAL TWIN IS A VIRTUAL REPRESENTATION OF REAL-WORLD ENTITIES AND PROCESSES, SYNCHRONIZED AT A SPECIFIED FREQUENCY AND FIDELITY.
(SOURCE: HTTPS://WWW.DIGITALTWINCONSORTIUM.ORG/GLOSSARY/INDEX.HTM#DIGITAL-TWIN)

The definition is further explained[7]:

- Digital twin systems transform business by accelerating holistic understanding, optimal decision-making, and effective action.
- Digital twins use real-time and historical data to represent the past and present and simulate predicted futures.
- Digital twins are motivated by outcomes, tailored to use cases, powered by integration, built on data, guided by domain knowledge, and implemented in IT/OT systems.

[6] The Asset Administration Shell also supports RDF. Since the Asset Administration Shell does not primarily focus on the definition of semantics but on digital twins it also supports technologies like XML, JSON, AutomationML and OPC UA.
7 https://blog.digitaltwinconsortium.org/2020/12/digital-twin-consortium-defines-digital-twin.html

4 How to solve the problem

4.1 Conceptual Architecture

In order to solve the challenges associated with contextualizing asset telemetry, an architecture that focuses on ingesting the time series data from assets and storing them in queryable repositories must be supplemented with a means to contextualize the asset data. The following diagram divides the solution space into a data environment, where data are stored, and a modeled environment where contextual objects are instantiated.

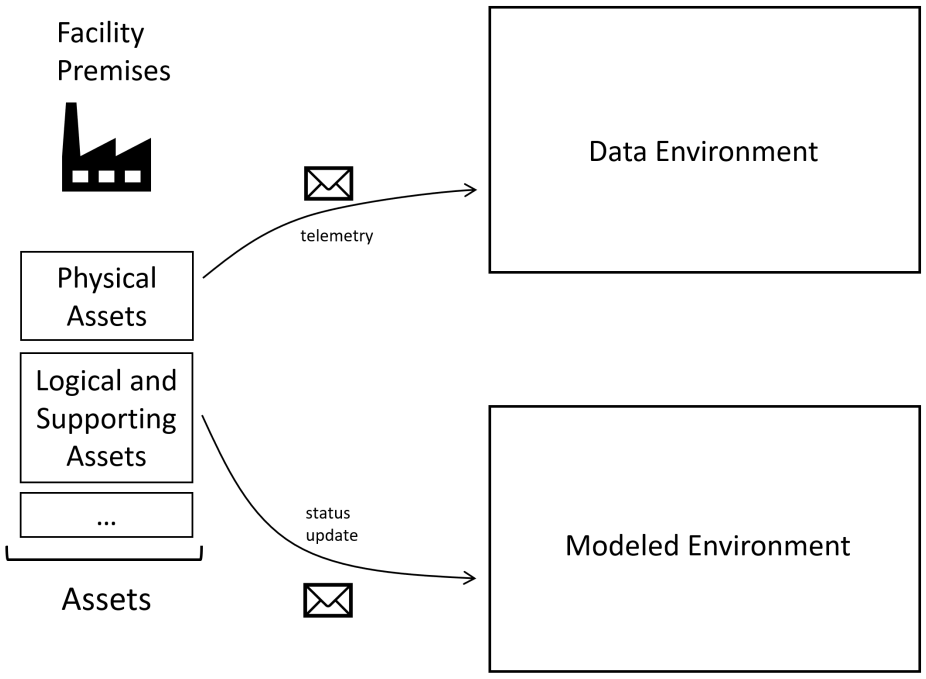

Figure 4-1 Conceptual Architecture

4.1.1 Facility Premises

As depicted above, the facility premises can contain multiple physical assets relating to devices that participate in a manufacturing process. In addition, logical and supporting assets are a combination of physical assets that are related and are often looked at as one telemetry generating asset. This asset telemetry can be transmitted in several ways.

4.1.2 Data Environment

The data environment collects, stores, and federates all data, being operational coming from all the connected assets (e.g., sensors, machines, PLC, …) and transactional coming from Enterprise systems (e.g., ERP, MES, PLM, CRM, …).

The data environment is composed of a plurality of technical components, like data lakes, time-series database, relational database (RDBMS), document-oriented database, object stores, triplestore, … The data environment may leverage the infrastructure provided by a cloud platform that offers the required compute, network, and storage resources.

This cloud can be a hyperconvergence of resources that exist in an organization's on-premises data center or hosted in a public cloud provider.

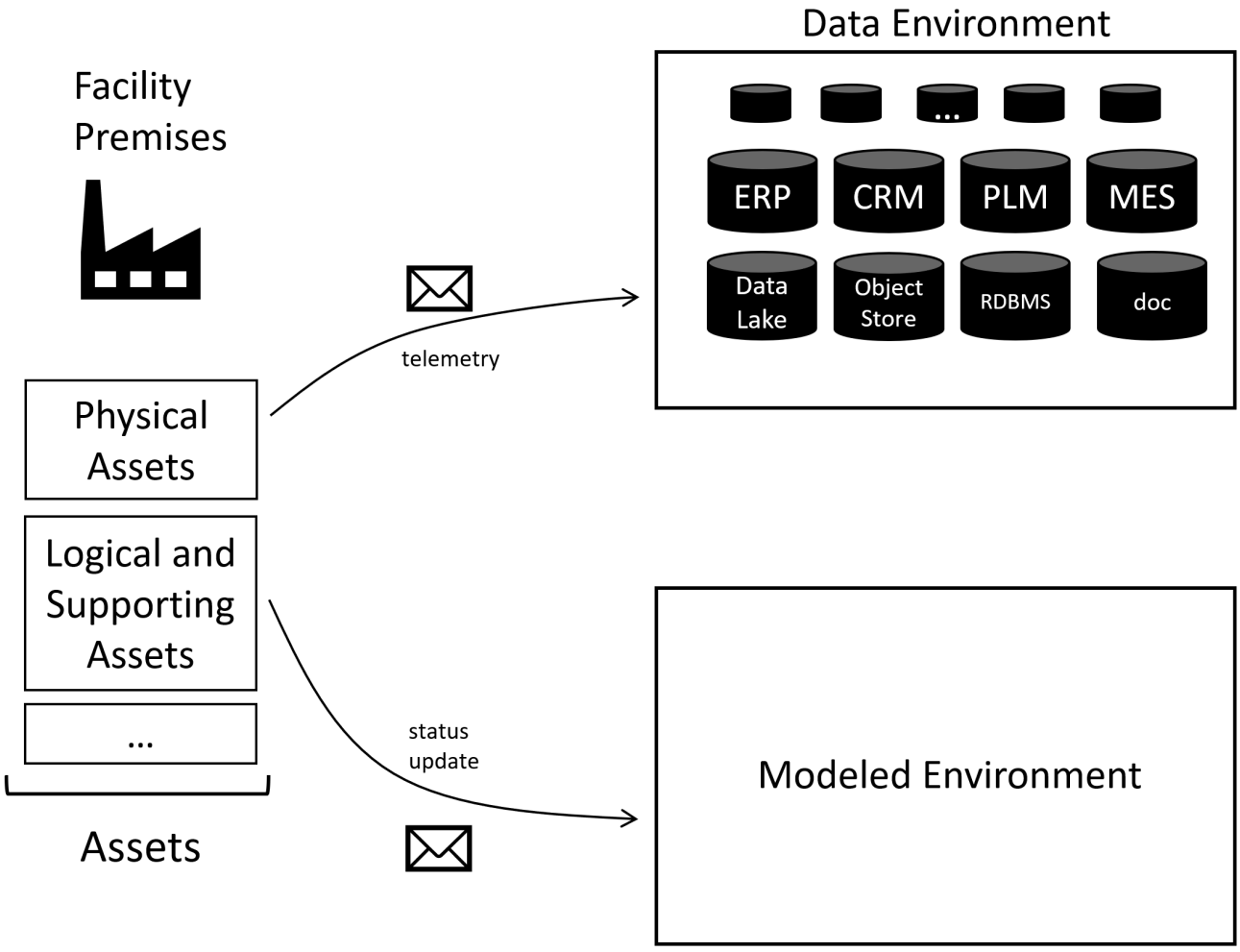

Figure 4-2 Data Environment

4.1.3 Modeled Environment

The modeled environment contains the digital twins with its instantiated aspects corresponding to the connected assets, like physical assets, logical and supporting assets, or others. The modeled environment contains context and details (e.g., serial number, OEM id, …) that the asset telemetry doesn't have on its own. The digital twin has the current state of the asset telemetry in its context (e.g., device temperature update).

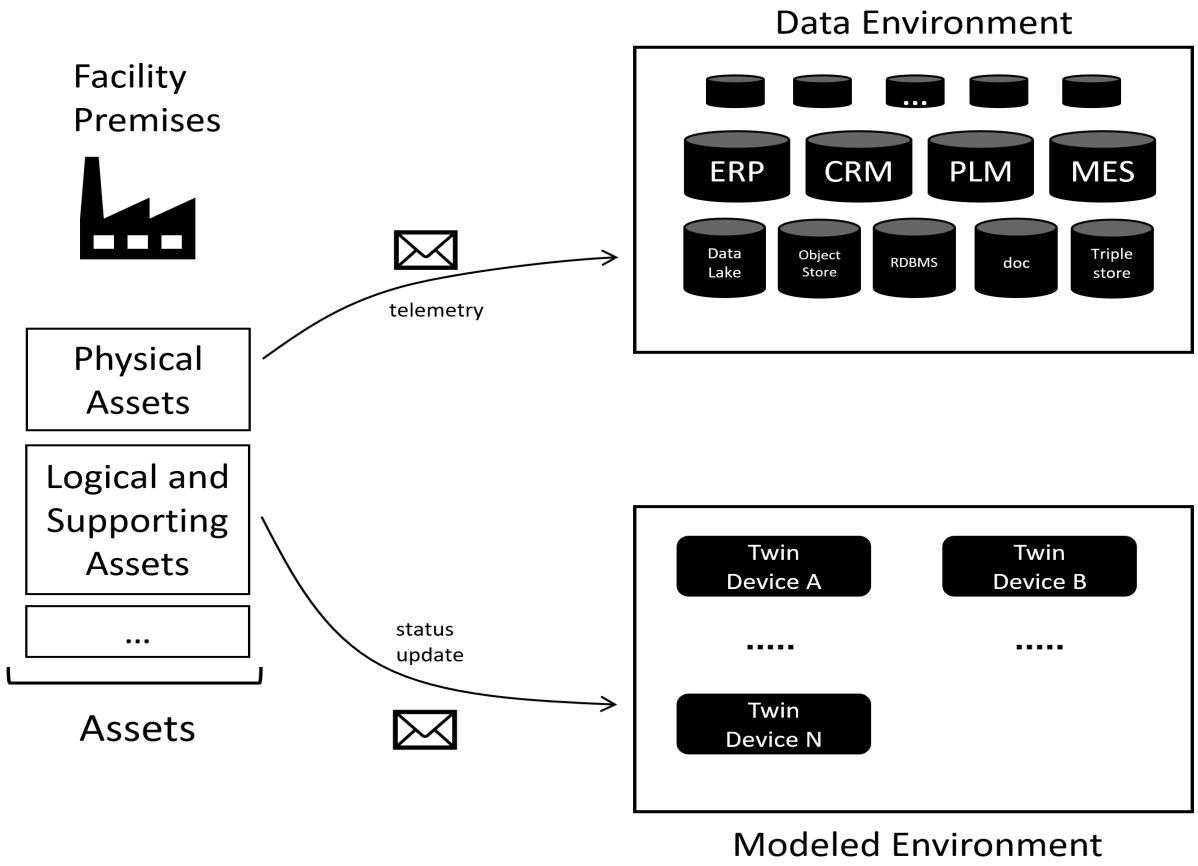

Figure 4-3 Modeled Environment

At this stage, two challenges need to be addressed:

- To maintain the digital twin models over time, we need a Semantic Data Model: The Aspect Meta Model (see section 4.1.4)
- To query, join and process data from the data environment together with the digital twins of the modeled environment: the Query & Analytics Platform (see section 4.1.5)

4.1.4 Aspect Meta Model

The modeled environment contains context and details that need to be defined in a standardized way. We can then speak of semantic entities that can be applied so that context can be infused into – for example - the time series data. This context is in the form of models that represent the asset that originated the telemetry. These models help users understand the assets, their relationships and then help with the query for time series data. These models will reside in a model store and will participate in the querying process.

A digital twin consists of a set of aspects, each aspect based on a well-defined semantic model or just aspect model (in our example, asset models like the Bill of Material or a Device Model) describing the information the application gets access via the API of the digital twin. As changes happen to assets, twins and models are adjusted to reflect reality within the facility. All aspect models follow a common meta model, the Aspect Meta Model. For details on the Aspect Meta Model see chapter 4.3.1.

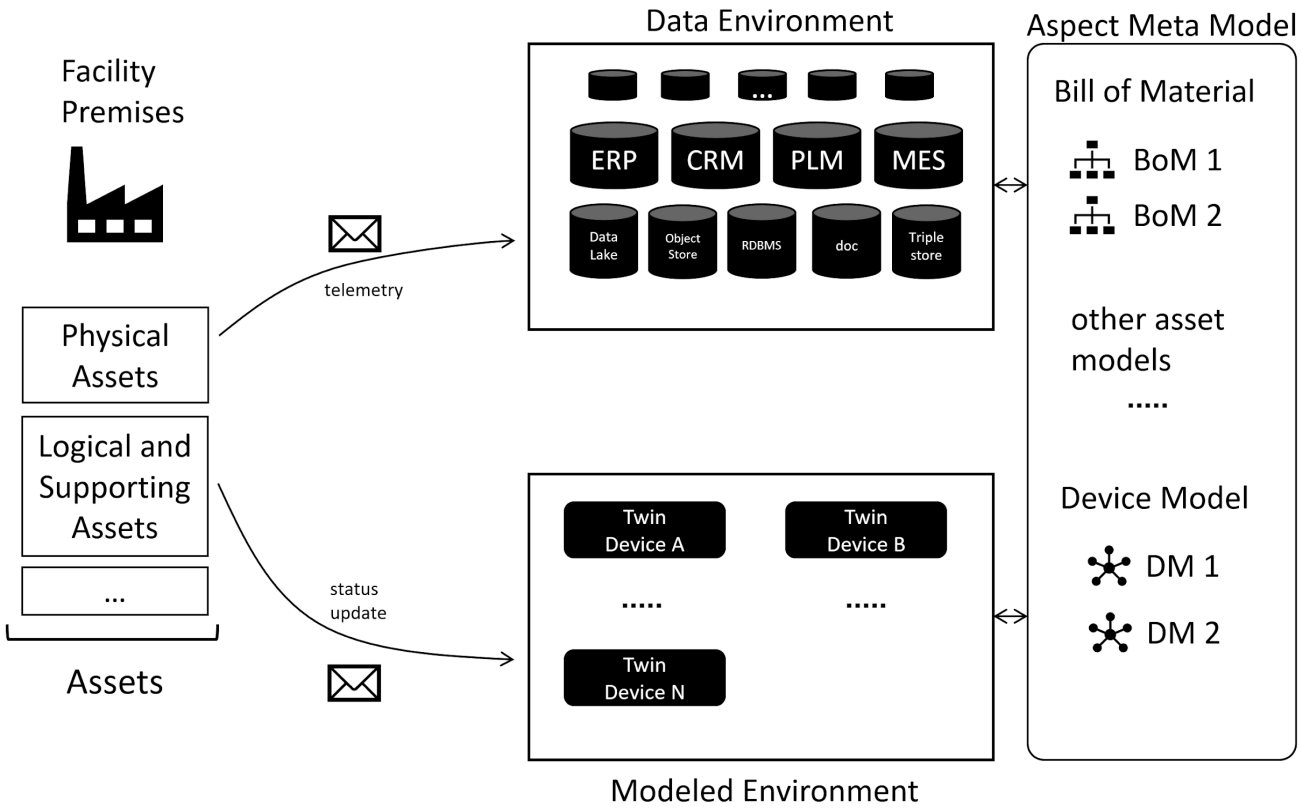

Figure 4-4 Aspect Meta Model

4.1.5 Query & Analytics Platform

The Query & Analytics Platform aims to federate all the data stored in multiple repositories of the Data Environment and "leverages/calls" the digital twins of the Modeled Environment to address use cases.

The Platform offers data processing capabilities that generate new data being turned into standard formats (e.g., SPARQL) or feeding new databases (business tables, semantic data layer, OLAP, …) to be exposed for consumption to users, applications through connectors (e.g., BI/dashboarding tools, …) or APIs (e.g., HTTP/REST).

For example, the Query & Analytics Platform allows for a seamless join of the telemetry data that contains the battery level of an AGV to the ERP data that contains its working plan for the next hours.

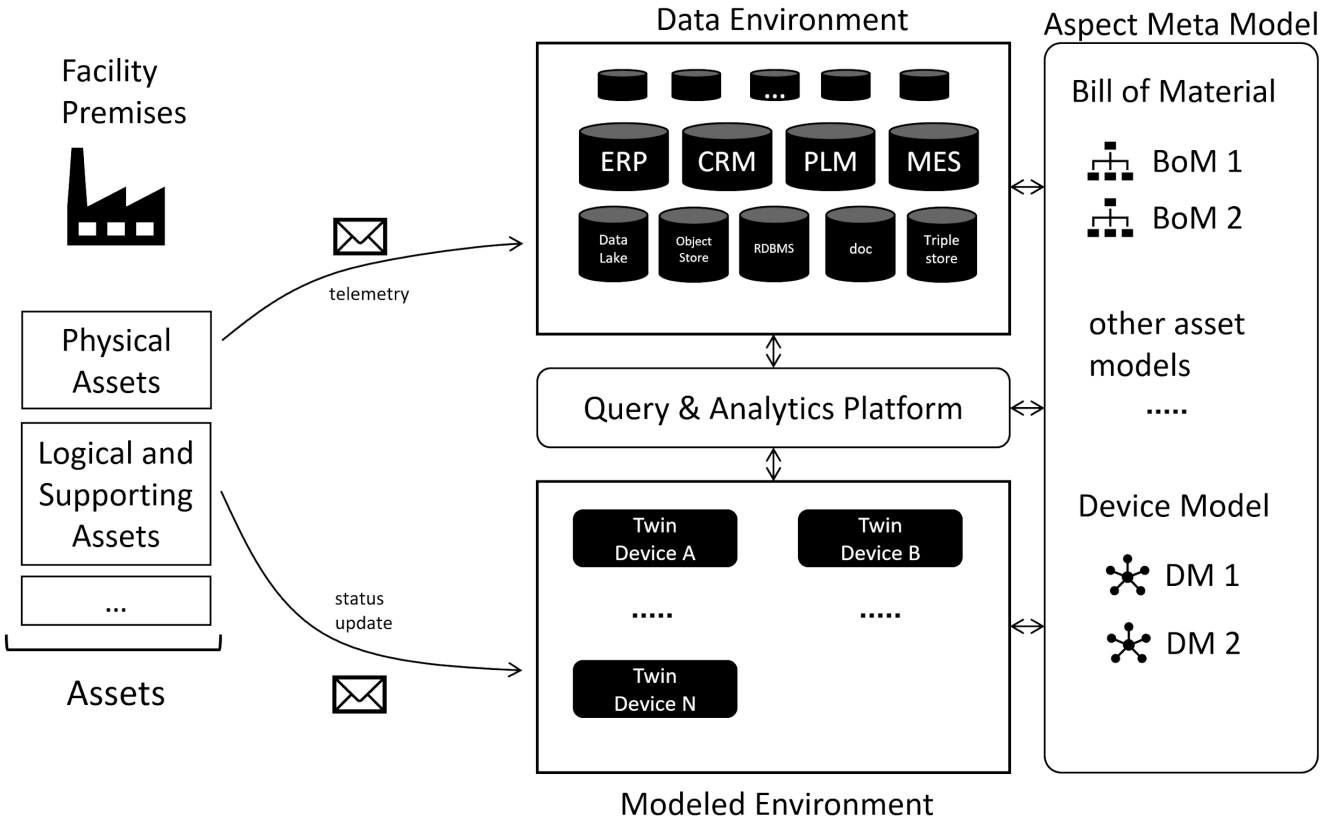

Figure 4-5 Query & Analytics Platform

4.1.6 Benefits of the Conceptual Architecture

The contextualization of asset data with models allows for the combination of IT and OT data for analytics. The semantic models, when applied to asset telemetry, make the data discoverable by users who wish to join OT time-series aggregate data with other data sources from other systems. Assuming that other systems such as ERP, CRM, or custom data sources similarly use a semantic framework, users can combine these data sets together to query.

For example, analytics such as “Tell me the average temperature and maintenance records of all boilers with model number 3654 in the Taichung, Taiwan plant for the last three weeks” become

possible. In this example, historical aggregations on time series data, twin information about the location and type of assets and maintenance data kept in a separate database are all combined in a single query.

4.1.7 Example based on Conceptual Architecture

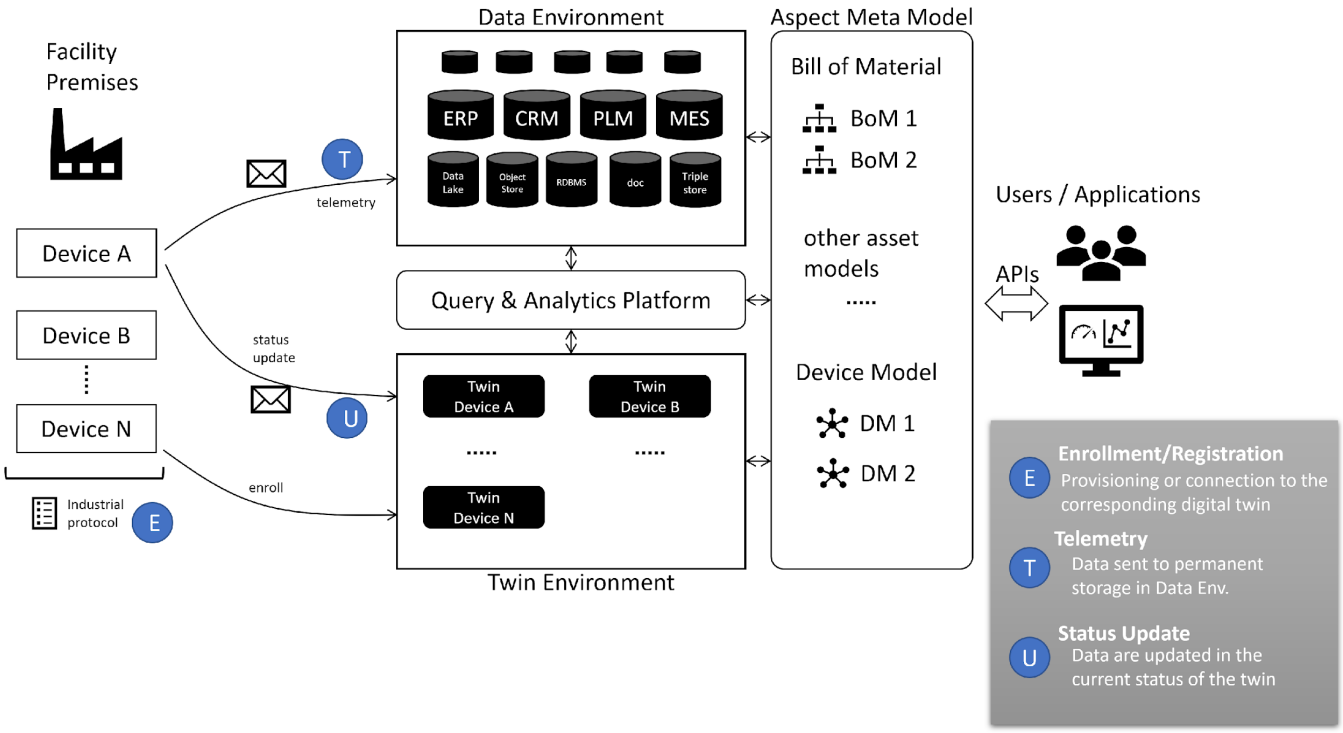

Figure 4-6 – Example for Usage of Conceptual Architecture

Here is a simplified example considering a specific asset, e.g., a welding robot supplied by two different manufacturers or of different models/versions. The semantic data model makes possible the definition of a single digital twin that embraces the diversity of assets having the same function. Therefore, the materialization of the data performed by OT does not depend on manufacturer choices. The access and the understanding of the data are ensured by a trusted digital twin that allows for a consistent materialization of the asset state at a given point in time, in a JSON file for instance. Over time, JSON files are generated, stored, and organized in the data environment, e.g., in an object store with an appropriate structure to simplify data queries. These data are then consumed by an analytic platform, that gathers data lakes, data warehouses, analytic engines, and many services dedicated to big data manipulation and processing. The analytic platform aims to merge the data of the data environment with other data sources, e.g., ERP or CRM, to provide all the required context to properly address the business question. The generated business insights are then exposed to applications through APIs or

dashboard tools with appropriate connectors to close the loop of the decision processes in providing the most relevant and explicit information to the business users/operators or directly to machines/assets to automate corrective actions.

4.2 Digital Twins

The hypotheses in chapter 3 showed that digital twins are the means to solve the problems sketched in chapter 2. In the previous chapter, we had a look at the overall conceptual architecture. So now, let's have a closer look at how a twin should look like to fulfill the hypotheses.

We already stated that a digital twin consists of a set of different aspects. Solutions can access the information provided by these aspects via a standardized interface. It is a very modular and flexible approach: aspects can be added and removed at any time during the life cycle of an asset.

An aspect groups together a set of related information about an asset to be consumed by a digital twin solution. An aspect comprises an aspect implementation and an aspect model.

An aspect implementation can be shared by several twins. There also can be a single implementation for several different aspects. This is illustrated in Figure 4-3.

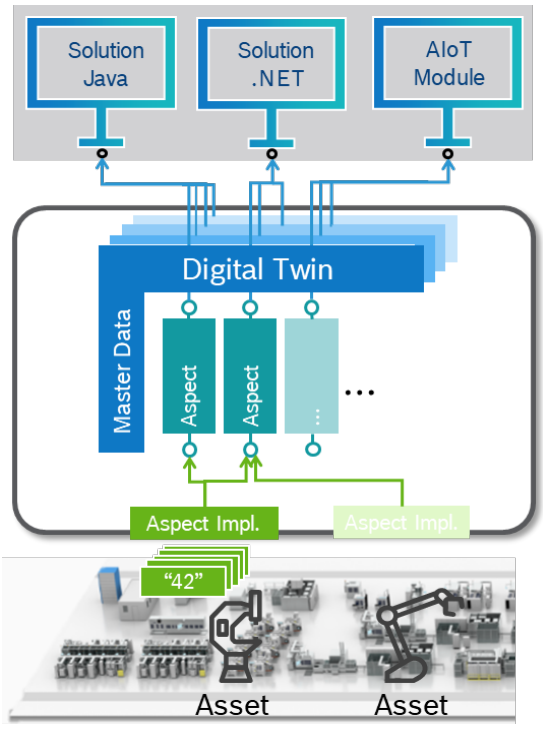

Figure 4-7 Digital Twin with its Aspects © Bosch

4.3 Semantic Framework for Digital Twins

The main building blocks for a semantic framework are sketched in Figure 3-8[8]:

- Semantic Meta Model
- Semantic Models with model implementations
- Semantic Runtime Environment including management, organization, and discovery of semantic information for specific assets
- Data Consumption
- Data Adaptation

These building blocks are described one by one in the following subchapters. With respect to the Semantic Runtime Environment the focus will be on *Registry* and the API for Solutions. This overview is not exhaustive: other management and organization services may be implemented. Additionally, Tooling is described in chapter 4.6.

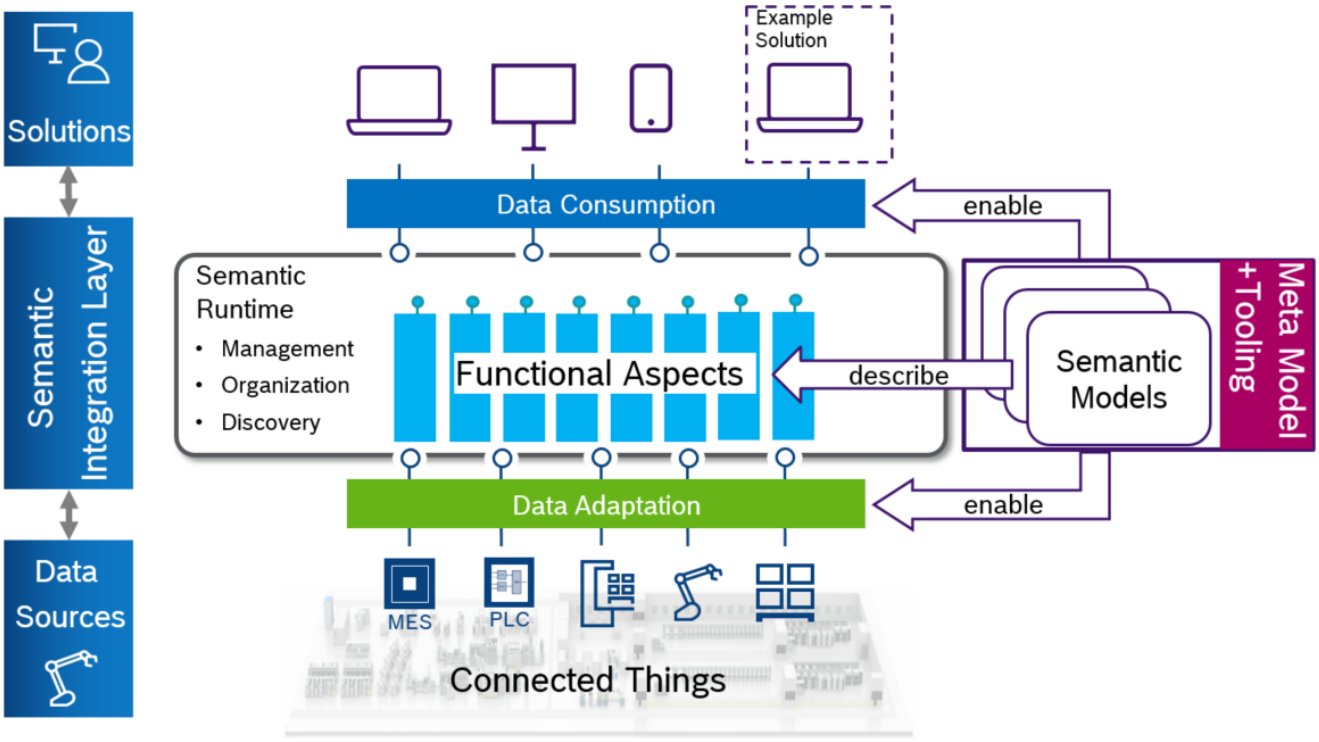

Figure 4-8 Building Blocks of Semantic Framework for Digital Twins[9]

[8] see also" Data Homogenization in the Age of Industry 4.0 - Digital Twin System as the basis for ease of communication between software, hardware and people in highly efficient production processes", July 2019, Bosch Connected Industry

[9] https://open-manufacturing.org/wp-content/uploads/sites/101/2021/06/OMP-WG-Semantic-Data-Structuring-Charter-V4_0_1-9-Sep-2020.pdf

4.3.1 Semantic Meta Model

There are three levels of abstraction:

- Meta Model (we call it semantic aspect metamodel - defines how to create valid Aspect Models)
- Model (we call it semantic aspect model, Structure Runtime Data & enrich it with semantics)
- Model Implementation (runtime, provide access to data for consumers)

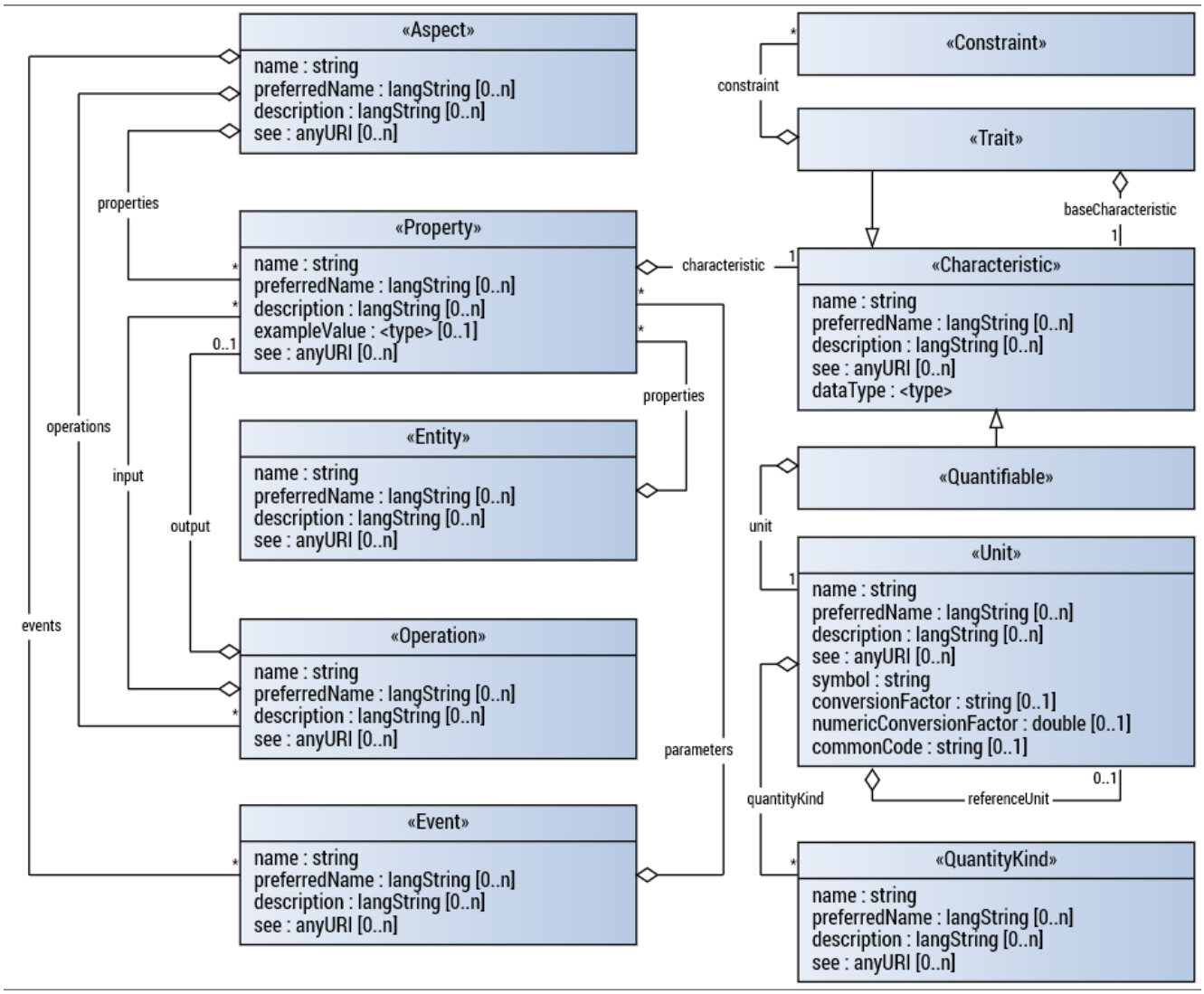

Figure 4-9 BAMM Aspect Meta Model

In this chapter, we deal with the metamodel. An aspect consists of properties and/or operations. The input and output parameters of an operation are treated as properties as well.

The semantics of a property is defined by a so-called characteristic. Properties for whom the value needs a physical unit to be correctly interpreted are called quantifiable. A Quantifiable is nothing but a characteristic with an additional semantically uniquely defined physical unit. For details, refer to the BAMM Aspect Meta Model Specification[10].

The Meta Model is specified using the Resource Description Format (RDF[11]) and the Terse RDF Triple Language syntax (TTL[12]), together with validation rules in the Shapes Constraint Language (SHACL[13]). These are defined in the scope of the "Semantic Web" initiative by the W3C.

4.3.2 Semantic Models with Implementation

The Semantic Model is an implementation of the previously defined Semantic Meta Model. The number of Semantic Models grows incrementally with use cases.

As an example, let's focus on the AGV use case described in section 2.1.3. and a corresponding digital twin approach described in paragraph 4.3. For this use case, two aspect models are defined:

- for the aspect "Battery" describing the battery level and the warning level,
- for the aspect "VehicleState" describing the state of the vehicle.
- The following properties are attached to aspect "Battery"
- the battery level of charge in % (Measurement *Characteristic*)
- the warning level with the three possible values: red, yellow, and green (*Enumeration Characteristic*)
- To ease maintenance a structuring property "Charging State" with an *Entity* is introduced, thus additional properties can easily be introduced for future versions of the aspect model

For the "VehicleState" aspect we just have a single property defined in our example:

- the operation mode (busy, idle, charging or error)

[10] https://openmanufacturingplatform.github.io/sds-bamm-aspect-meta-model/bamm-specification/v1.0.0/index.html

[11] Richard Cyganiak, David Wood, Markus Lanthaler. RDF 1.1 Concepts and Abstract Syntax. W3C Recommendation, 25 February 2014. URL: http://www.w3.org/TR/rdf11-concepts/

[12] Eric Prud'hommeaux, Gavin Carothers. RDF 1.1 Turtle: Terse RDF Triple Language. W3C Recommendation, 25 February 2014. URL: http://www.w3.org/TR/turtle/

[13] Holger Knublauch, Dimitris Kontokostas. Shapes Constraint Language (SHACL). W3C Recommendation, 20 July 2017. URL: https://www.w3.org/TR/shacl/

Although not part of the example aspects additional aspects may be modelled, for example an aspect contains an operation like:

- goToChargingStation() that is triggered when the battery warning level is yellow and the VehicleState is either idle or busy.

The aspect models are shown with their definition using the BAMM Aspect Meta Model in the following (except for the name spaces to ease reading, for details refer to the BAMM Aspect Meta Model Specification[14].

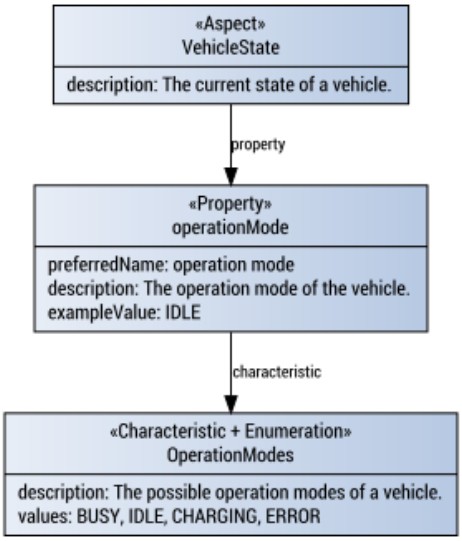

Figure 4-10 Aspect Model for Vehicle State of an AGV

```
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix bamm: <urn:bamm:io.openmanufacturing:meta-model:1.0.0#> .
@prefix unit: <urn:bamm:io.openmanufacturing:unit:1.0.0#> .
@prefix bamm-c: <urn:bamm:io.openmanufacturing:characteristic:1.0.0#> .
@prefix bamm-e: <urn:bamm:io.openmanufacturing:entity:1.0.0#> .
@prefix : <urn:bamm:com.example:0.0.1#> .

:VehicleState a bamm:Aspect ;
    bamm:name "VehicleState" ;
    bamm:properties (:operationMode) ;
    bamm:operations () ;
    bamm:description "The current state of a vehicle."@en .

:operationMode a bamm:Property ;
    bamm:name "operationMode" ;
    bamm:description "The operation mode of the vehicle."@en ;
    bamm:preferredName "operation mode"@en ;
```

[14] https://openmanufacturingplatform.github.io/sds-bamm-aspect-meta-model/bamm-specification/v1.0.0/index.html

```
    bamm:exampleValue "IDLE" ;
    bamm:characteristic :OperationModes .

:OperationModes a bamm-c:Enumeration ;
    bamm:name "OperationModes" ;
    bamm:description "The possible operation modes of a vehicle."@en ;
    bamm:dataType xsd:string ;
    bamm-c:values ("BUSY" " IDLE" " CHARGING" " ERROR") .
```

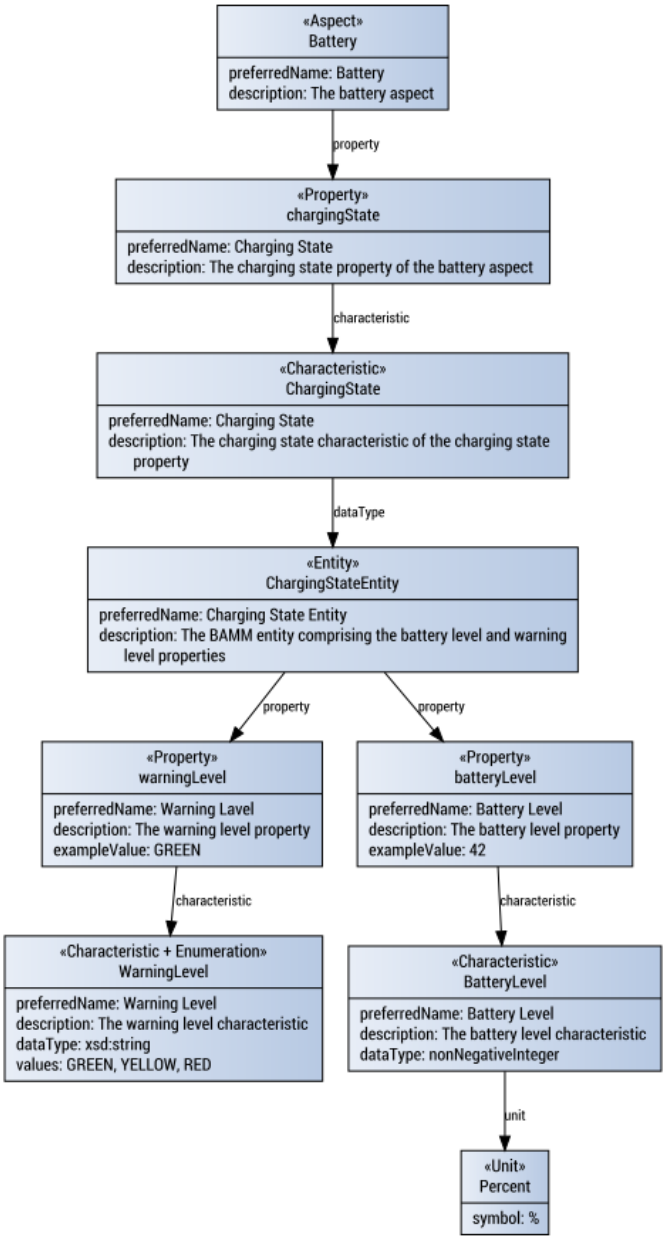

Figure 4-11 Aspect Model for Battery of an AGV

```
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix bamm: <urn:bamm:io.openmanufacturing:meta-model:1.0.0#> .
@prefix unit: <urn:bamm:io.openmanufacturing:unit:1.0.0#> .
@prefix bamm-c: <urn:bamm:io.openmanufacturing:characteristic:1.0.0#> .
@prefix bamm-e: <urn:bamm:io.openmanufacturing:entity:1.0.0#> .
@prefix : <urn:bamm:com.example:0.0.1#> .


:Battery a bamm:Aspect ;
    bamm:name "Battery" ;
    bamm:preferredName "Battery"@en ;
    bamm:description "Aspect that represents battery level and warning level of the
AGV."@en ;
    bamm:properties (:chargingState) ;
    bamm:operations () .

:chargingState a bamm:Property ;
    bamm:name "chargingState" ;
    bamm:preferredName "Charging State"@en ;
    bamm:description "The charging state of the battery"@en ;
    bamm:characteristic :ChargingState .

:ChargingState a bamm:Characteristic ;
    bamm:name "ChargingState" ;
    bamm:preferredName "Charging State"@en ;
    bamm:description "charging state characteristic"@en ;
    bamm:dataType :ChargingStateEntity .

:ChargingStateEntity a bamm:Entity ;
    bamm:name "ChargingStateEntity" ;
    bamm:preferredName "Charging State Entity"@en ;
    bamm:description "The BAMM entity comprising the battery level and warning level
properties"@en ;
    bamm:properties (:batteryLevel :warningLevel) .

:batteryLevel a bamm:Property ;
    bamm:name "batteryLevel" ;
    bamm:preferredName "Battery Level"@en ;
    bamm:description "The battery level property"@en ;
    bamm:exampleValue 42 ;
    bamm:characteristic :BatteryLevel .

:warningLevel a bamm:Property ;
    bamm:name "warningLevel" ;
    bamm:preferredName "Warning Lavel"@en ;
    bamm:description "The warning level property"@en ;
    bamm:exampleValue "GREEN" ;
    bamm:characteristic :WarningLevel .

:WarningLevel a bamm-c:Enumeration ;
    bamm:name "WarningLevel" ;
```

```
    bamm:preferredName "Warning Level"@en ;
    bamm:description "The warning level characteristic"@en ;
    bamm:dataType xsd:string ;
    bamm-c:values ("GREEN" " YELLOW" " RED") .

:BatteryLevel a bamm-c:Measurement ;
    bamm:name "BatteryLevel" ;
    bamm:preferredName "Battery Level"@en ;
    bamm:description "The battery level characteristic"@en ;
    bamm:dataType xsd:nonNegativeInteger ;
    bamm-c:unit unit:percent .
```

For every aspect model, there also needs to be an aspect implementation. The implementation can be reused to realize a set of digital twins providing the same aspect.

4.3.3 Semantic Runtime Environment

4.3.4 API for Solutions

Solutions want to access the information provided via aspects of a digital twin in a harmonized way. The basis for generating the payload of such APIs are the aspect models introduced in the previous sections. In the following, we look at our example aspect model “VehicleState”.

This is the payload schema that is directly derived or even automatically generated from the aspect model as defined before:

```
{
  "$schema" : "http://json-schema.org/draft-04/schema",
  "type" : "object",
  "properties" : {
    "operationMode" : {
      "type" : "string",
      "enum" : [ "BUSY", " IDLE", " CHARGING", " ERROR" ]
    }
  },
  "required" : [ "operationMode" ]
}
```

The payload at run-time of a query asking for the corresponding Vehicle State Aspect then could look like this at a specific point in time:

```
{
   “operationMode” : “BUSY”
}
```

4.3.5 Registry

It is of no use in the semantic data framework that there is a digital twin providing relevant information if the solutions interested in the data do not know about these digital twins or do not know how to access them. This is why there is a need for a registry service. The registry service allows a user or solution to find all digital twins providing the aspect data they are interested in. The registry provides metadata together with endpoint information on how to access the data via a standardized API. Of course, not every user or every solution is allowed to access all aspects of digital twins. Here access control comes into play.

4.3.6 Discovery

A similar problem as with digital twins and the information they provide already arises when setting up a new digital twin. In doing so, there is the need-to-know which aspect models and which aspect implementations are already available, and which need to be defined or implemented from scratch. For helping to discover existing aspect models with their implementations there is the need to define a catalog containing the aspect models and implementations but also providing an API that allows querying the models from the perspective of a solution.

For an aspect modeler, it is the other way around: they want to make available the aspect model and implementation and then make it public via the catalog API.

4.4 Semantic Data Adaption

When creating or adapting an aspect model, the most direct way is to modify respective Aspect Model RDF files. However, this requires a deep understanding of modeling in general and, more specifically, of the underlying Aspect Meta Model. This contradicts the idea of including domain-specific knowledge and semantics into the aspect model because the domain experts are often not very familiar with such modeling approaches, and they lack the time for deeper training. As a consequence, the Aspect Meta Model becomes more useful, and its adoption can be increased when there is good tooling for the adaptation of the semantic information in the aspect models.

As a foundation for this tooling, developing a set of SDKs in different languages like Java or .Net Core is useful. Among other things, such an SDK could implement the validation of aspect models regarding their conformance to the Aspect Meta Model or provide semantic model adaptor skeletons.

Especially for more complex models, it is often also easier to have visual representation which requires viewers for the models. Such a viewer becomes even more beneficial when using it as an editor to modify and create aspect models.

4.5 Semantic Data Consumption

To actually get a benefit out of the modeling of the semantics for data points, other systems should, of course, consume and make use of this information. Examples for such consuming applications could be among others, user interfaces for the current state of an asset or solutions for performing analytics on the data. The starting point in each case is to enable the consuming systems to retrieve the models and find the digital twins and data sources that implement the aspects defined in the models. This can be done through a registry which is explained above.

If there is no semantic model available, as is the case in most scenarios as of today, the developer needs to directly hard code the semantics into the specific implementation. This requires that the developer is either also an expert for the domain or in close contact with such an expert during the whole development phase. It also makes it difficult to adapt the application if parts of the semantics change.

A showcase of what a consuming system can then do with the additional semantic information can be based on our AGV example. For the following, we assume that an application wants to display the state of multiple AGVs that drive around a shop floor. This application then queries a digital twin registry or similar database for the available AGV data. However, this data is not sufficient to know what the values actually mean and how to display them in the user interface. Without an aspect model, the developer would then need to directly hard code the semantics into the implementation. The application would fetch the aspect models to get this semantic information. Without further involvement from the developer, the application could then automatically render the user interface. For the Vehicle State aspect of the AGV, this could be a text field with the data value for the operation mode that is annotated with the label for `preferred name` and a mouse-over hint with the `description`. The `preferred name` and the `description` are then both extracted from the aspect model. For the battery level, there could be a status bar that becomes easier to render since the user interface already knows the value range from the semantic model.

This consumption of the semantics requires additional tooling to make it as seamless as possible for the application developer. Otherwise, the benefit of having the same semantic description over the whole life cycle of the data may be nullified by the effort of consuming the aspect model. Through the definition of an Aspect Meta Model, this consumption step becomes more independent from the context of the actual application and can therefore be bundled into a library. Since a larger number of different applications can rely on it, such a library or SDK is needed.
However, it would not directly cover the consumption of the actual data. This is dependent on the provider of the data and needs to be solved for each data source which could be, for example, a digital twin system, a database, or other data brokers.

In many scenarios, the owner of the data further wants to differentiate the level of access for the different consumers. Describing the asset through different aspect models already leads to a

separation of the relevant and undividable aspects of an asset. In that regard, adding a role-based (or even attribute-based) access control system to the data source, which manages the access to the data on the level of the different aspects make sense.

For our AGV example, a possible scenario is that operator of the plant owns the AGVs and needs the vehicle state and the battery information for internal route planning. At the same time, the owner may have a service contract with an external company for the maintenance of the AGVs. As this external company only needs the information that an AGV is in an error state for a longer time, the owner could only grant access to the “VehicleState” aspect but not the data associated with the “Battery” aspect.

4.6 Semantic Tooling

Making good use of the aspects model and ease their creation requires good tooling along the whole life cycle of the models. As already outlined above, two important topics are the adaption and the consumption of the models. This involves the development of editors and libraries, which make it easy to interact with the models.

At the same time, the list of useful tooling can be extended depending on the use case. For example, another useful tool would allow to set up new digital twins. The user would use such a tool to create digital twins by assigning them, specific aspect models. In a second step, the user connects the digital twin with a data source that implements the aspect or at least provides data that can be used to implement the aspect. The target audience for this and further tooling may explicitly be domain experts who need to focus on their regular tasks and are therefore not familiar with the whole system architecture.

By relying on the same meta model, such tooling can also be kept rather generic, which allows the applicability to multiple use cases and contexts.

5 How to start/Getting started

This section describes how to start this work within an organization in order to get benefit from the value of a Semantic Framework. The overall prerequisite is to work on the modeling of the assets under consideration in the overall system (model) and solution space. The rationale to do so is the business use cases the organization has in mind.

There are two ways to start; either by defining the ontology of the complete system, which will require strong expertise in that field, or by introducing the new BAMM Aspect Meta Models according to the needs of the business use cases under consideration. The BAMM Aspect Meta Model is a proven model which covers the key concepts of semantic modeling.

To make the model relevant and valid, it also requires some business domain knowledge. The objective here is to gather the overall understanding domain experts can bring together to map it to the system

model. These experts may come from engineering, manufacturing, planning, logistics domains bringing strong expertise both from a use case standpoint and a business data value. The idea is to consolidate the different viewpoints in the Semantic Framework while making the relationships between them explicit. The challenge is to stay at the business level and not to dive into IT system modeling at that time. The model of the system should be technology agnostic and be able to describe the use cases of the business domains.

It is also crucial to consider scalability from the beginning: The approach sketched in the paper can be used for a small number of digital twins but is scalable up to several million digital twins.

6 What will come next

This whitepaper started by describing problem symptoms and the hypotheses that digital twins are a feasible way to solve these problem symptoms.

We introduced a conceptual architecture for a semantic framework distinguishing between data and modeling environment. The digital twins are part of the modeling environment. With our running example of an automated transport system (AGV) we showed how to set up a digital twin with two aspects for battery level and vehicle state and also how to access the information via a standardized API. Also, registration and discovery of aspect models in a catalog were considered for scalability.

In the future whitepapers, we will also examine how to apply semantics and semantic data structuring to the edge or even the device (machine) to better address machine operator or machine builder problems.

Another important topic to address is asset and model evolution, including how to react to changes in the asset facilities and how to incorporate changes in the models without interrupting production. For example, a new software update of a machine may support a new version of the OPC UA Companions Specification it supports. Or there is a new version of an aspect model such as how to refactor the aspect implementation and how to support applications that used the former version of the aspect model.

Homogenization is crucial in such a context with many stakeholders from different companies in such an ecosystem. The Open Manufacturing Platform enables standardization via its open-source approach. The BAMM Aspect Meta Model is already open to the public. In the next steps, additional open-source code tooling will be provided to enable the building of an open-source community dedicated to semantics and digital twins. An alignment with other activities within OMP like the Manufacturing Architecture WG and the Connectivity WG will fill the gaps in the overall picture.

Of course, the Open Manufacturing Platform is open to existing standards and looks for synergy between approaches as taken by the Industrial Digital Twin Associations, GAIA-X, CATENA-X or ECLASS – to name only a few.