OPEN MANUFACTURING
PLATFORM

Industry 4.0 Core Information Model

A publication by the Open Manufacturing Platform

Draft
2022-05-11

www.open-manufacturing.org info@open-manufacturing.org

OPEN MANUFACTURING
PLATFORM

Legal Disclaimers

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

AUTHORS (in alphabetical order)
Soufiane Ameziane (Teradata Corp.)
Irlan Grangel-Gonzalez (Robert Bosch GmbH)
Piotr Janik (Intel Corp.)
Marcel Keller (Robert Bosch GmbH)
Felix Loesch (Robert Bosch GmbH) – Coordinator of Whitepaper I4.0 Core Information Model
Fabio Massari (VMware Inc.)
Denis Molin (Teradata Corp.)
Nico Wilhelm (ZF Friedrichshafen AG)

CONTRIBUTORS
Birgit Boss (Robert Bosch GmbH)

Contents

1 Scope

1.1 Introduction / Objectives

The objective of Industry 4.0 is to leverage all available data to get the most accurate understanding of how the business went in the past, how it goes in the present and how it will likely be in the future – so that the best decisions can be taken at the right time. In other words, it aims to solve business questions with data-driven insights.

Typically, business users address business questions through applications that connect to and process all the relevant data. These applications act as an interface between the business users and the insights extracted from the available data and relevant to the business question to solve.

1.2 Status Quo

The data, collected through digitization processes, is typically stored and made available in a diverse set of fragmented and isolated silo systems, using technology depending on the nature, lifecycle, and volume of these systems.

Very often, the collected data – referred to as raw data - do not directly or straightforwardly answer business questions, or at least not the questions that bring the most value to the business. The data require a set of transformations to be presented in an understandable and actionable form so that specific business questions can be addressed successfully.

Transformations could be logical such as basic combinations, aggregations, and filtering but also more advanced, like machine learning, deep learning, or graph analytics - or they could be technical, like a change of file system or topology. This would help simplify querying, ensure scalability and concurrent usage of the same data, and meet specific service level agreements, typically driven by the business pace.

Transformations require certain compute resources that could be embedded in the applications or shared across multiple applications in a query & analytic platform. The products of raw data transformation typically require storage resources to foster the reuse that could be handled by the raw data storage system or in dedicated storage resources like object stores, relational databases, or triple stores.

1.3 Main Difficulties / Problems

The main difficulty faced by the industry is to find, localize and access the right data, be it raw data or already available transformed data, to address the problem at hand. This is especially true when dealing with a fragmented ecosystem resulting from a long history of mergers and acquisitions or a diversity of specialized activities in which data is spread over large incompatible systems involving different technology.

1.4 Solution – The Core Information Model

This guidance is exactly the role played by the core information model approach presented in this white paper describing the data and their relationships (see Figure 1). Through a semantic layer, the core information model maps the business ontology with all the available data – raw or transformed – often stored in a technical manner to meet technical constraints. The role of the core information model democratizes and simplifies the access to serve the needs of the business in developing analytics and application in a trusted and intelligent manner.

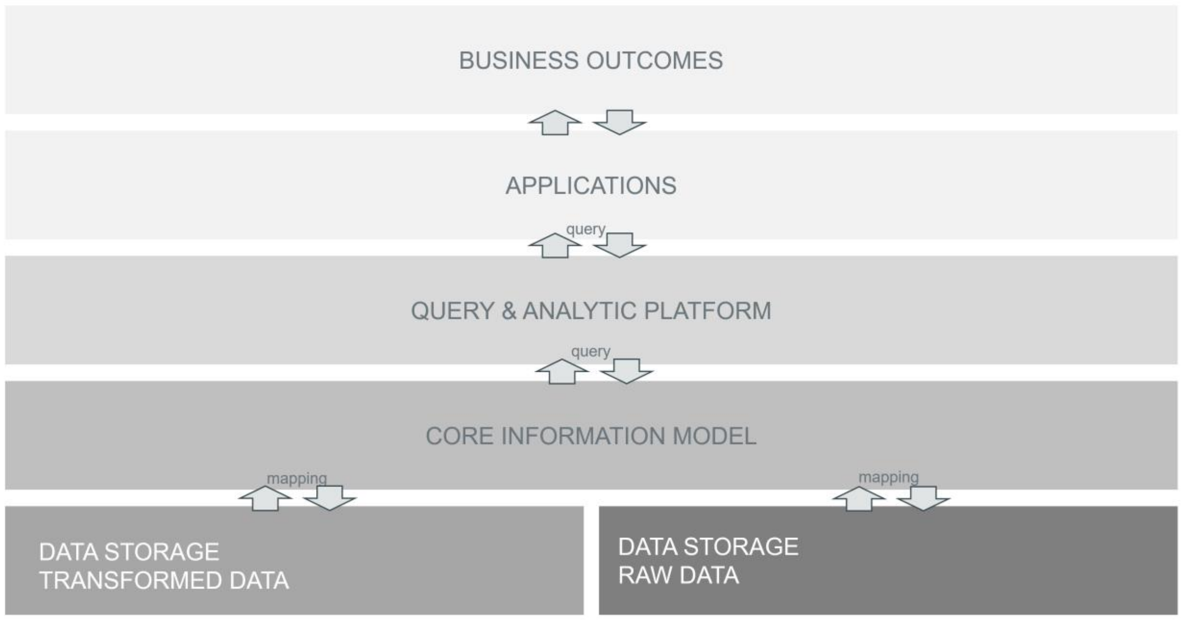


Figure 1: Role of Core Information Model in the Data Architecture[1]

2 Motivation

Currently, a great effort is required to acquire, access, find, integrate, describe, and harmonize manufacturing data residing in heterogeneous and isolated data silos before it can be used effectively in data processing applications (e.g., for KPI reporting, quality analysis, process optimization or other I4.0 use cases). Furthermore, data is very often integrated in an ad-hoc and application-specific way, and the data integration logic is hard-coded, which results in very low reusability of the developed applications (see Figure 2).

[1] See also the OMP Whitepaper "Reference Architecture", online https://open-manufacturing.org/wp-content/uploads/sites/101/2021/05/OMP_Reference_Architecture_Whitepaper-17-May-21.pdf

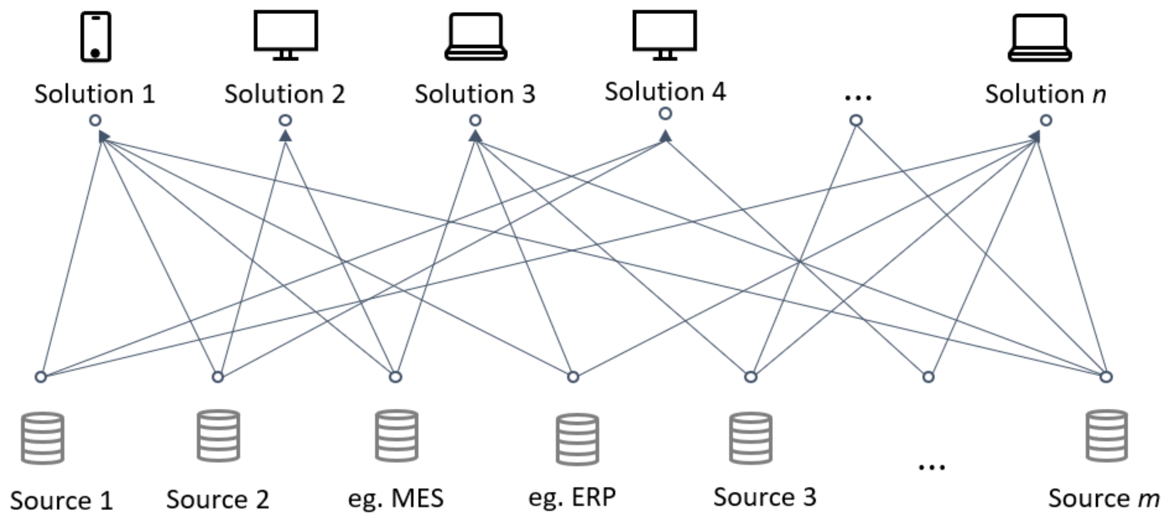


Figure 2: Status Quo - Application-specific, hard-coded integration of data with high effort

According to various studies, 80% of the effort is spent on data preparation tasks, whereas only 20% is spent on data analytics (see Figure 3). This is effectively an inferior data strategy.

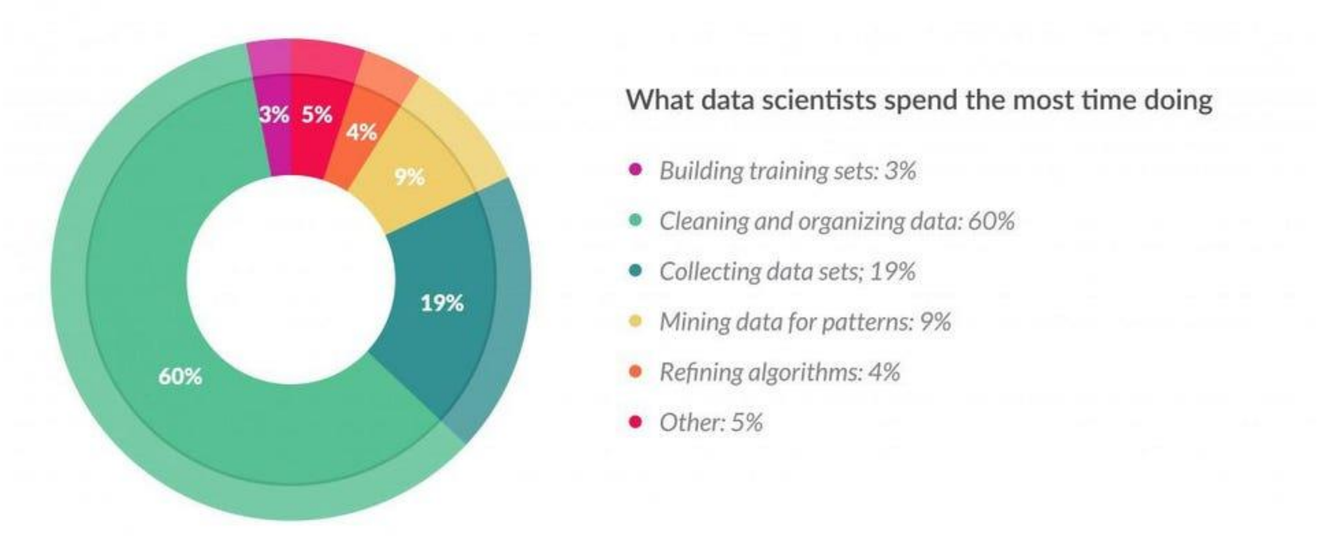


Figure 3: 80% Effort for Data Preparation, 20% Effort for Data Analysis (Source: Crowd Flower Data Science Report 2016)

To overcome this and enable the most efficient use of data in I4.0 applications, the following challenges need to be addressed[2]:

Heterogeneity of Data Sources

- Data is typically stored in heterogeneous data sources, isolated silos, in different formats (e.g., relational database 1..n, CSV, XML, JSON, …), and following different schemas.
- Due to the heterogeneity of data sources and the involvement of many stakeholders, so-called semantic interoperability conflicts are very common in the data:
    - In different data sources, different syntactic names for semantically identical business terms/objects are used (e.g., Material vs. Product)
    - In different data sources, identical syntactic names for semantically different business terms/objects are used (e.g., Process vs. Manufacturing Process and Business Process)

Lack of Standardization

- Currently, there is a lack of standardization regarding relevant terms, entities, their attributes, and relationships in the manufacturing domain.
- Although some effort has been put into defining standards for the data entities in the manufacturing domain (e.g., ISA 95, IEC 62264), these standards are not implemented as machine-readable information models, nor are they adopted widely in the manufacturing industry.

Lack of Semantic Context

- Manufacturing data is often stored without metadata and without context. This prevents the understanding of the data by different stakeholders as the meaning and context of data need to be reconstructed by the data consumer instead of being available right with the data
- Very often, there is a lack of descriptive information and metadata. For example, cryptic names are used for columns in databases, or possible values of enumerations are not clear.
- There is currently no standardized way or data model of describing data in the manufacturing domain

Tight coupling of Data Processing Applications with Data Sources

- Data Processing Applications are directly developed against data sources, and the integration logic is hardcoded into the application. This makes it very hard to change the integration logic if new data sources or data producers (e.g., new machines) have to be integrated into the data processing applications.

[2] See also the OMP whitepaper "Semantic Data Structuring", online: https://open-manufacturing.org/wp-content/uploads/sites/101/2021/09/OMP-Semantic-Data-Structuring-Whitepaper.pdf

- The tight coupling furthermore results in very poor reusability of the applications across production lines, plants, products, and companies
- Data might be lost or created because of demand, but fictitious data might be introduced in the application-specific hard-coded integration logic

Data not ready for Utilization or Application of AI

- Data cannot be directly consumed from heterogeneous and raw data sources as the data sources do not provide the meaning of data
- Very often, significant effort is needed to understand, prepare, and transform the data before AI algorithms can be applied to it

Lack of Data Sharing

- Data sharing is prevented by isolated data silos and fear of leakage of the company-specific IP in the data.

Heterogeneity of Machines and Practices in Multiple Plants

- Machines on the shop floor are typically purchased from different suppliers and are therefore heterogeneous in terms of the data they provide, even if they execute the same manufacturing process.
- Practices in multiple plants are also diverse due to a long history of mergers & acquisitions.
- Data projects typically start in one plant and are later rolled out in different plants, with slight changes leading again to heterogeneous data implementations.

3 Solution Approach – I4.0 Information Model

To address the challenges mentioned in Section 2 with respect to missing common understanding of relevant terms and their relationships, we propose the Industry 4.0 Core Information Model as a standardized conceptual data model in the manufacturing domain. The Industry 4.0 Core Information Model serves as a semantic abstraction layer between the heterogeneous data sources and the data processing applications (see Figure 4).

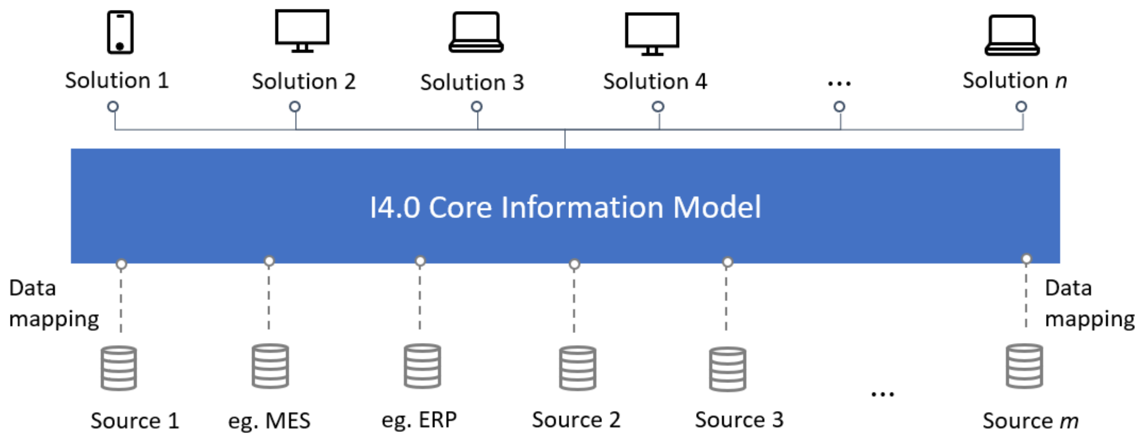


Figure 4: Solution Approach – Industry 4.0 Core Information Model as Semantic Abstraction Layer

The I4.0 Core Information Model enables the following benefits:

1. **Standardization of business terms used in manufacturing:** The I4.0 Core Information Model follows the International Standard IEC 62264 for Enterprise-Control System Integration and proposes standardization of relevant business terms and entities which are provided the shared semantic meaning for all stakeholders in the manufacturing domain. It furthermore enables efficient communication between different stakeholders as terms are precisely defined.
2. **Semantic Context:** Furthermore, the I4.0 Core Information Model provides additional semantic context as the meaning of all entities, their attributes, and relationships between entities are clearly defined. This enables the data processing applications to ask complex queries that require following relationships between data elements, e.g., which sub-parts were supplied by supplier A and assembled in products of category B and produced in production line C by machines of machine vendor D.
3. **Decoupling of Data Processing Applications from underlying Data Sources**: The I4.0 Core Information Model enables the harmonization of heterogeneous data sources and serves as a standardized data model for data processing applications, thus decoupling the applications from the underlying data sources.
4. **Enabling the development of reusable data processing applications:** Due to the standardization via the I4.0 Core Information Model, data processing applications can be developed once against the I4.0 Core Information Model and then simply be reused in other production lines, across plants, or even across business units without re-developing them from scratch.
5. **Enabling reuse of transformed, derived, or conformed data:** The I4.0 Core Information Model manages both raw and transformed data. Data are of two kinds: data that are present in a source system that is typically accessed by a read operation, and data that are the results of a transformation from its basic state into something that is useful from a business or an analytic viewpoint (a very basic example: in a resistance welding process, the current and voltage are raw data, that is materialized, whereas the resistance is the ratio of voltage and current and is not

necessary materialized in the source system, although it has a greater interest from a business viewpoint).

6. **Reduction of effort required for data preparation:** As a consequence of the standardization of entities, attributes, and their relationships by the I4.0 Core Information Model and the enablement of development of reusable data processing applications, the effort required for data preparation is significantly reduced.

4 General Concepts of Core Information Models

4.1 Overview

In this section, we present the general concepts of core information models. We start with the definition of a core information model: *"A core information model in software engineering and data modeling is an abstract, formally represented conceptual data model expressing concepts, attributes and relationships, constraints, rules, and operations to specify data semantics for a chosen domain of discourse. A core information model thus defines the blueprint or schema for data. Data or data instances are usually not part of an information model."*

In this whitepaper, we use the means provided by semantic modeling languages such as RDF, RDFS, and OWL to express the concepts, attributes, and relationships. Compared to other modeling languages, semantic modeling languages are particularly suited to convey the meaning of instances described in terms of the defined concepts, attributes, and relationships. They can express and exchange information, which enables different parties (e.g., stakeholders, business units, and different companies) to interpret the meaning of the instances.

4.2 Semantic Concepts

To model the *entities*, *attributes,* and *relationships* of a specific domain, the semantic modeling languages (e.g., OWL) offer the following means. In the semantic language OWL, an entity is called a class, an attribute is called a data property, and a relationship is called an object property:

- *Classes*: Classes in a semantic model are used to describe abstract *entities*. For example, if one wants to describe the entity *machine*, the concept of a *class* can be used.
- *Data properties*: Data properties are used to describe *attributes* of entities. For example, the attribute "serialNumber" of the entity "Machine" is described using a data property
- *Object properties*: Object properties are used to describe relationships with a defined semantic meaning between two classes. For example, the relationship "belongsToBuilding" can be described by an object property between the classes "Machine" and "Building".

4.3 Model as Blueprint vs. Instance Data

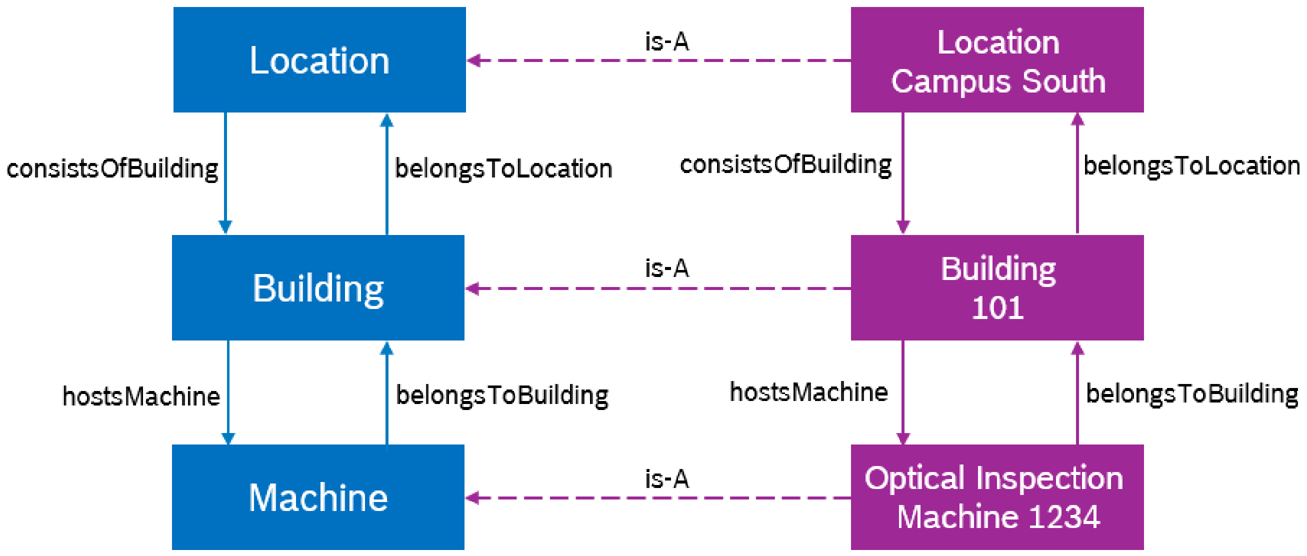


Figure 5: Distinction between Classes and Instance Data -
Classes are shown in blue and data instances are shown in purple.

Figure 5 shows a simple example of a semantic core information model which consists of the classes "*Location*", "*Building*" and "*Machine*" (shown in blue) and the relationships "*belongsToLocation*", "*belongsToBuilding*", "*consistsOfBuilding*" and "*hostsMachine*" (shown as solid blue arrows). In Figure 5, these classes are instantiated with concrete data (shown in purple). Each instance, e.g., "*Location Campus South*" is related back to the class, which defines the general concept (see dotted arrows labeled with "is-A").

This is a very powerful concept as instance data is not only just represented as data but is linked to its semantic meaning. As you can see from Figure 5, the classes and relationships in the model serve as a blueprint and semantic description of the meaning of the instances (purple part). The relationships (object properties) defined at the level of classes can be used between instances of the classes. This allows us to use the blueprint for a semantic description of instance data. By relating the instance data back to the classes, the interpretation of the data is clear to the data consumer. This is particularly beneficial in cases in which multiple stakeholders or parties need to have a common understanding, i.e., the same meaning of the data.

4.4 Modularization Concept

Semantic information models can be structured using so-called sub-models. A sub-model typically represents a specific part of the domain to be described in the semantic information model. Sub-models can be included by other models, thus providing a powerful concept for modularization. As one might want to relate classes defined in one sub-model with classes defined in another sub-model, these relationships need to be captured somewhere. Two different options for capturing these inter sub-model relationships can be distinguished:

- *Option 1 (Common Model)*: All inter sub-model relationships are defined and owned by the common model that includes the sub-models. This option has the advantage that the sub-models are confined and do not refer to other sub-models.
- *Option 2 (Direct Linking)*: Relationships between classes of sub-model A and sub-model B are defined in sub-model A. This has the advantage that you see these relationships in sub-model A. However, the disadvantage now is that sub-model A needs to include sub-model B. The more sub-models you have, the more complicated option 2 will get.

Due to the disadvantages of option 2, we propose using option 1 (Common Model) to implement inter-sub-model relationships. However, in order to better understand the relationships between the models in this white paper, we are showing the relationships to classes from other models directly in the sub-model. Relationships to classes from other models are shown in grey color in order to differentiate them from the classes defined in the sub-model currently being looked at.

5 The I4.0 Core Information Model

5.1 Overview

The I4.0 Core Information Model follows the International Standard IEC 62264 for Enterprise-Control System Integration. The IEC 62264 is a good starting point but does not fully address solving all of the above-mentioned challenges. Hence the I4.0 Core Information Model extends the IEC 62264 with additional classes, attributes, and relationships where needed to obtain a precise semantic model for the manufacturing domain and to address all of the challenges. For example, we added classes describing process parameters and result values needed to capture the process result data generated whenever a workpiece is being processed in a machine.

To describe the different areas of manufacturing, the I4.0 Core Information Model is divided into multiple sub-models. Each sub-model describes a specific area of the manufacturing domain. As in manufacturing, these areas are usually not separated but are related to other areas (e.g., a machine is producing a product or a machine is executing a manufacturing process). The classes described in one sub-model are linked to classes described in a different sub-model. In order to capture this, we have defined a common model which includes the sub-models and is used to define the relationships between classes from different sub-models.

The information model is available as an RDF ontology in the OMP SDS GitHub repository.

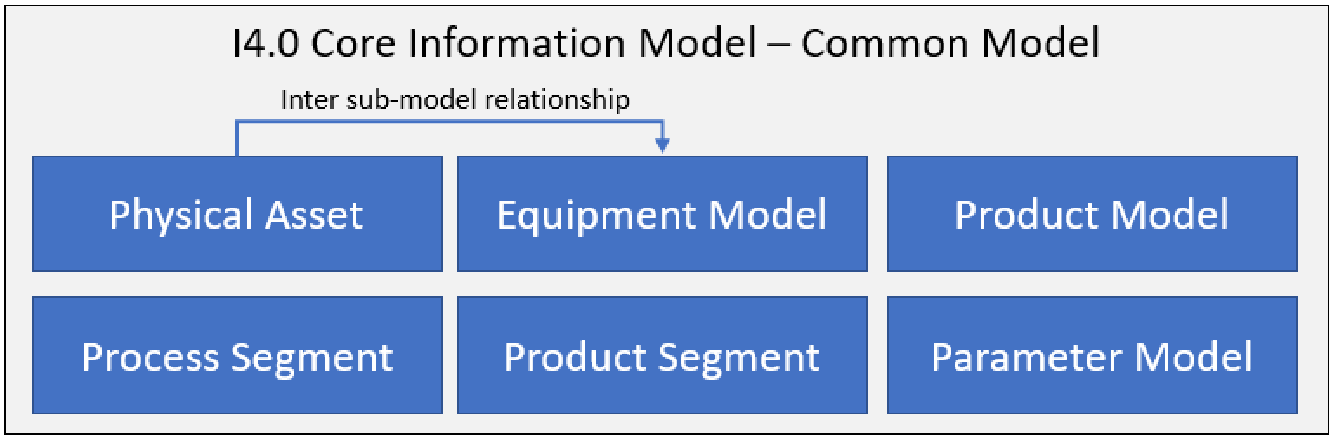


Figure 6: I4.0 Core Information Model - Common Model including Sub-Models

The I4.0 Core Information Model consists of the following sub-models in
order to describe the different aspects of manufacturing:

- *Physical Asset Model*: description of physical assets in manufacturing (e.g., location, building, machine, machine components, machine tools, …)
- *Equipment Model*: the content of the equipment model are classes to describe the logical structure of production (e.g., plant, production line, station, work unit)
- *Process Model:* description of manufacturing processes
- *Product Segment Model:* description of product segments
- *Product Model*: description of product structure (e.g., Bill of Material)
- *Parameter Model*: description of process parameters (e.g., setting, result parameters, product quality parameter, …)

Hint: Each of the classes in any sub-model of the I4.0 Core Information Model can be further specialized using the sub-class construct provided by RDFS / OWL. Thus, the I4.0 Core Information Model can be adapted to additional use cases which have not yet been foreseen in the model.

5.2 Information Model Case Example

To explain the concepts of the I4.0 Core Information Model, we will use an example of a production line through Sections 5.2-5.8. Figure 7 displays our exemplary production line. It consists of five conventional machines (dark grey squares), one transport band (blue bar), and three loading stations (light grey boxes). The production line produces transmission housings.

Raw alloy housing material (blue circles) usually depart from work cell 100.1 to the pre-machining work cells 10.1 or 10.2, which belong to Station 10 (Housing). The part is bolted to a special machining fixture. Once machined from several sides which are accessible, the part is transported to work cell 20.3. There, relocation on the fixture is required to enable complete machining from all sides. The work cells 20.1 and 20.2 complete the machining. For the final assembly process steps, the finished parts (green circles) are being arranged on carriers.
The work cell 30.1 serves as an access resource to adjust maintenance, staffing differences, or priority operations. Due to cutting speed differences, the work cell 20.3 is usually restricted to the repositioning of semi-finished parts and final part dismount.
This example is modeled as an RDF ontology available in GitHub.

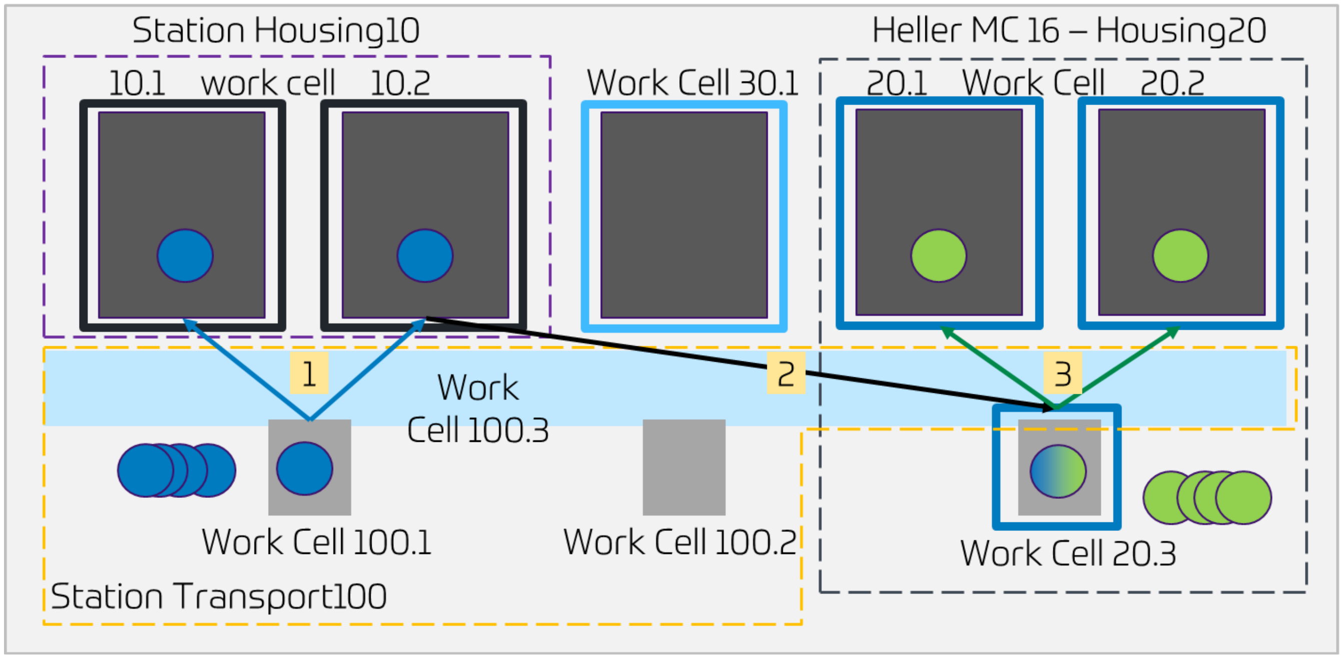


Figure 7: Example used to illustrate I4.0 Core Information Model

5.3 Physical Asset Model

The physical asset model describes physical assets used in manufacturing. A *physical asset* is a physical object uniquely identified and tracked for maintenance and/or financial purposes. Examples of physical assets in the manufacturing context are production machines, production tools, buildings, and locations. The physical asset model also describes important relationships between the main physical assets. For example, one can retrieve the hierarchical structure of physical assets via the "consistsOf" relationship. The physical asset ontology modeled in RDF is available in GitHub.

Figure 8 shows an overview of the physical asset model consisting of the following classes:

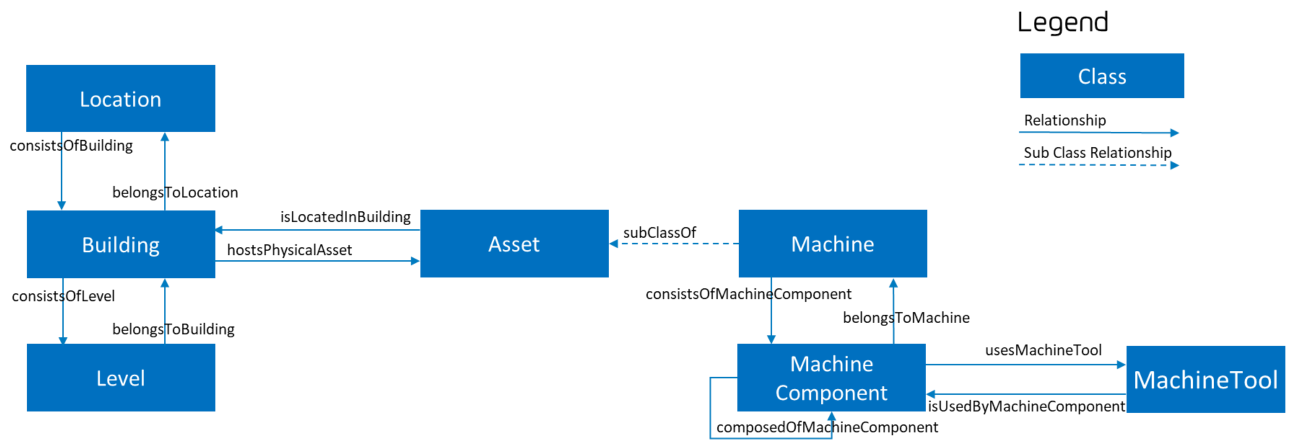


Figure 8: I4.0 Core Information Model – Physical Asset Model

Table 1: I4.0 Core Information Model – Physical Asset Model – Definition of Classes

Class	**Definition**
Location	Collection of open spaces and one or more buildings on defined properties (e.g., geo fenced). A spatial region or named place. Reference: http://purl.org/dc/terms/Location
Building	Independently usable, roofed structural works in a property. They represent a sheltered area to provide space for

	manufacturing assets, people and systems to perform value added activities.
Level	Sub structure of a building and is synonym to the word *building floor*
Asset (Production)	*Physical asset* that covers primarily machines and tools but excludes buildings. Note 1: Short form *asset* used in this document refers to *production asset*. Assets represent an extension point. Note 2: You may further subclass assets (e.g. AGVs, …) Reference: IEC 62264-1 ID:3.1.36, Industrie 4.0 – Terms, VDI status report Industrie 4.0 (April 2021)
Machine	A specific asset (production) subclass used in manufacturing or assembly to produce tangible goods.
Machine Component	A part of a machine that itself may be composed of sub-components.
Machine Tool	A device used in a machine for manufacturing or assembly. Note 1: For example, a drilling tool is used in a drilling machine to drill holes into the product.

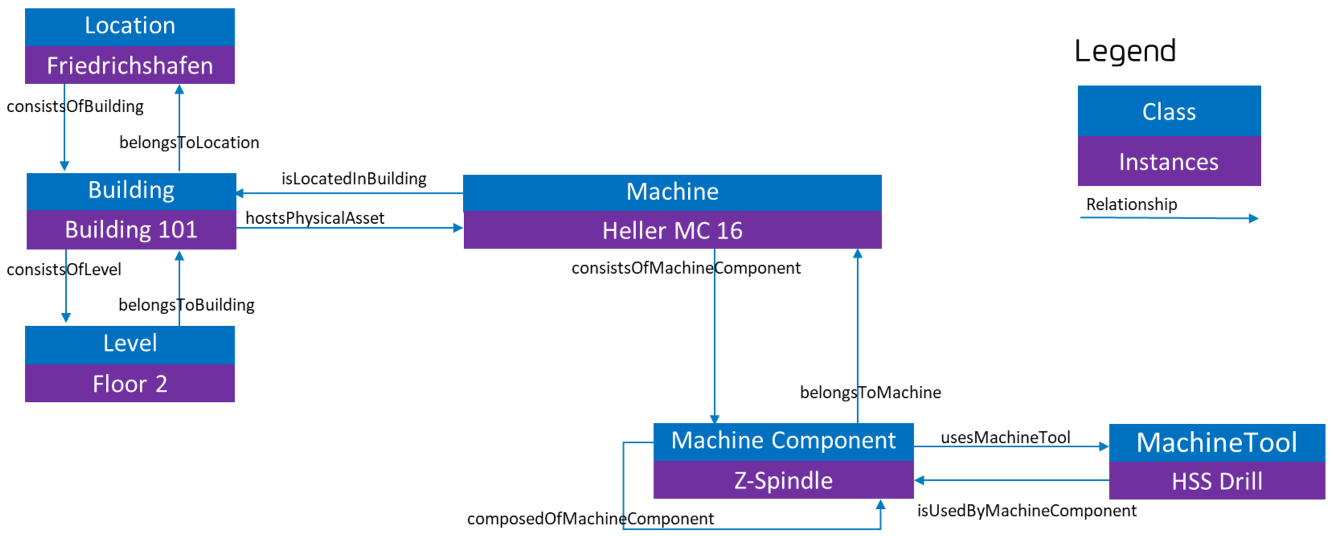


Figure 9: Physical Asset Model Example

5.4 Equipment Model

The equipment model describes a role-based equipment hierarchy, i.e., logical groupings of assets used for manufacturing. The differentiation of physical assets and equipment is based on the definitions in the IEC 62264-1 (see Figure 10).

There is a temporal relationship between instances of an equipment class (e.g., a station) and instances of a physical asset class (e.g., a machine) called *"isImplementedBy"*. This allows a physical asset (e.g., a machine) to implement two different work units. If the physical asset breaks down and a new one is bought, the relationship *"isImplementedBy"* needs to be adjusted, and the equipment (e.g., station 30) remains the same. The equipment ontology modeled in RDF is available in GitHub.

Figure 11 shows the equipment model. The classes shown in Figure 11 are defined in Table 2.

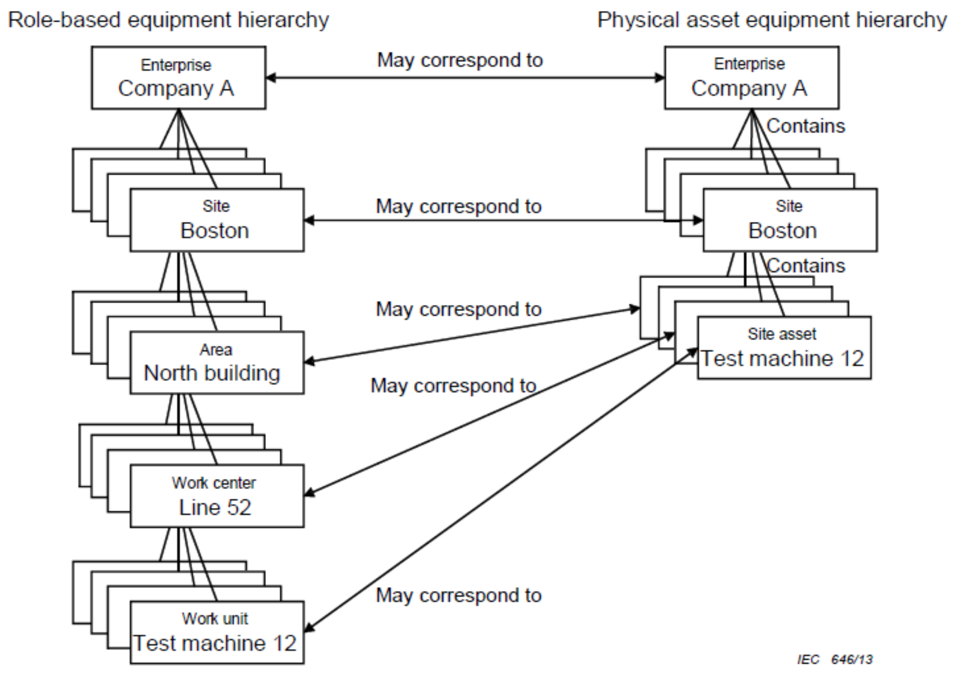


Figure 10: Equipment vs. Physical Asset – IEC 62264-1, Figure 6

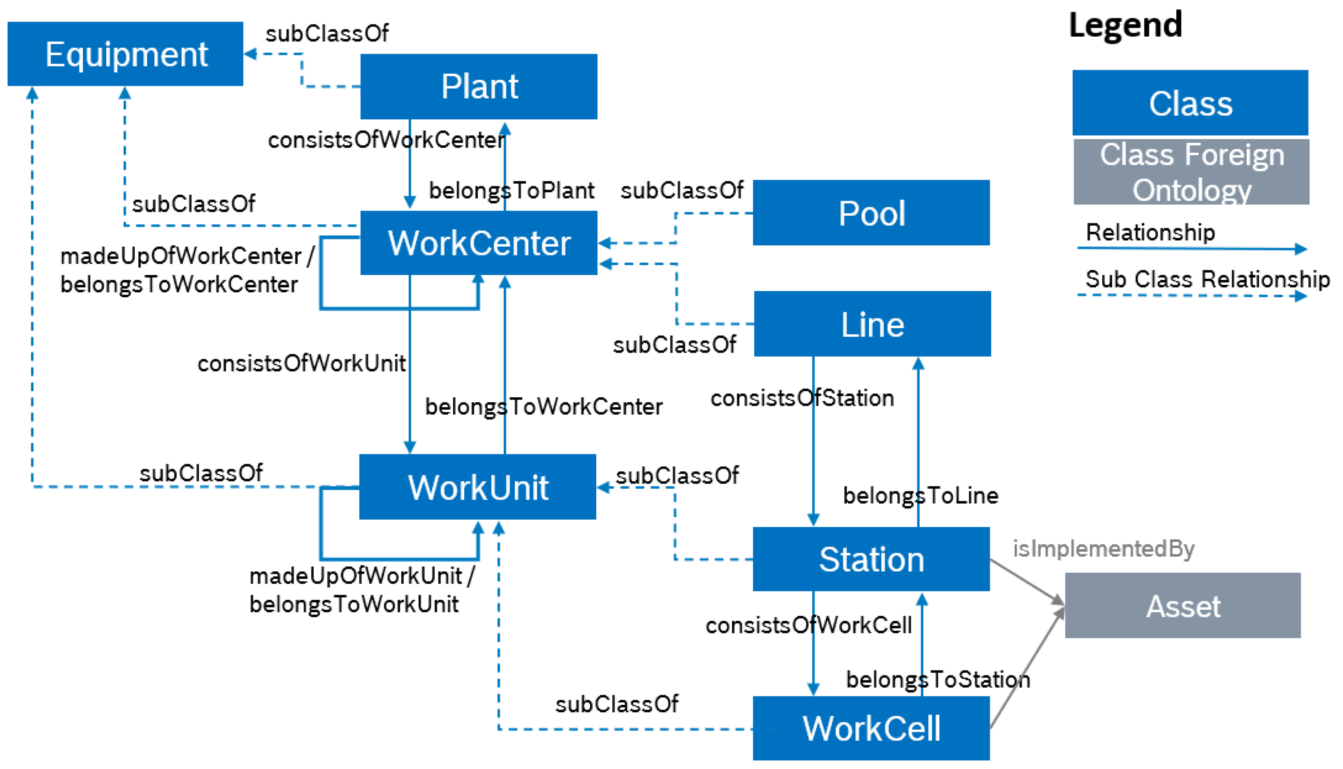


Figure 11: I4.0 Core Information Model - Equipment Model

Table 2: I4.0 Core Information Model – Equipment Model – Definition of Classes

Class	**Definition**
Equipment	Generalization for all logical elements required to organize or model your manufacturing hierarchy, sequence or flow. Note 1: The formal UML role-based equipment model object is used to define the role-based equipment hierarchy information that is defined in IEC 62264-1. Note 2: The model contains the information that may be used to construct the hierarchical models used in manufacturing scenarios. Reference: IEC 62264-1, IEC 62264-2
Plant	Complex and separate organizational structure set up. It is a logical grouping of a manufacturing entity. It contains other equipment such as work centers or work units. Plants may be part of business-oriented entities such as product lines, divisions, sales segments and so on.
Work Center	Grouping of one or more work units or smaller work centers that are logically or geographically linked. Line and Pool are sub-classes of work centers. A work centers can consist of 1..n work centers (e.g., a

	pool is part of a line). This term does not refer to SAP work center. It is possible but not recommended to represent an asset as a work center.
Pool	Grouping of several parallel work units running the same process or sharing similar capabilities. A pool is a subclass of a work center.
Line	Collection of one or more sequentially ordered work units that work together to perform a defined sequence of production processes to manufacture or assemble a product that can be a final or a semi-finished product. A line is a subclass of a work center.
Work Unit	Logical grouping of equipment at the level below a work center. A work unit can consist of smaller work units. A work unit logically represents the location of executing a manufacturing process and can be linked to a machine or machine component which physically executes the manufacturing process. It performs production, storage, material movement, or any other Level three or level four scheduled activity according to the manufacturing operations and control systems hierarchy. Reference: IEC 62264-1 ID:3.1.26
Station	A logical grouping of equipment intended to produce a good. Stations are often are linked to a manufacturing process for a semi-finished or finished part. Subclass of a work unit. Stations belong to a line. A line can consist of 1..n stations. One or more process segments may be linked to a station. Note 1: Assets or parts within an Asset represent a station well. Station can be linked to assets via the *"isImplementedBy"* relationship
Work Cell	Logical sub groupings of stations. A station can consist of 1..n work cells. Subclass of a work unit. Work cells are intended to implement a process segment. Usually, the mapping between process segment and work cell is a 1:1 mapping.

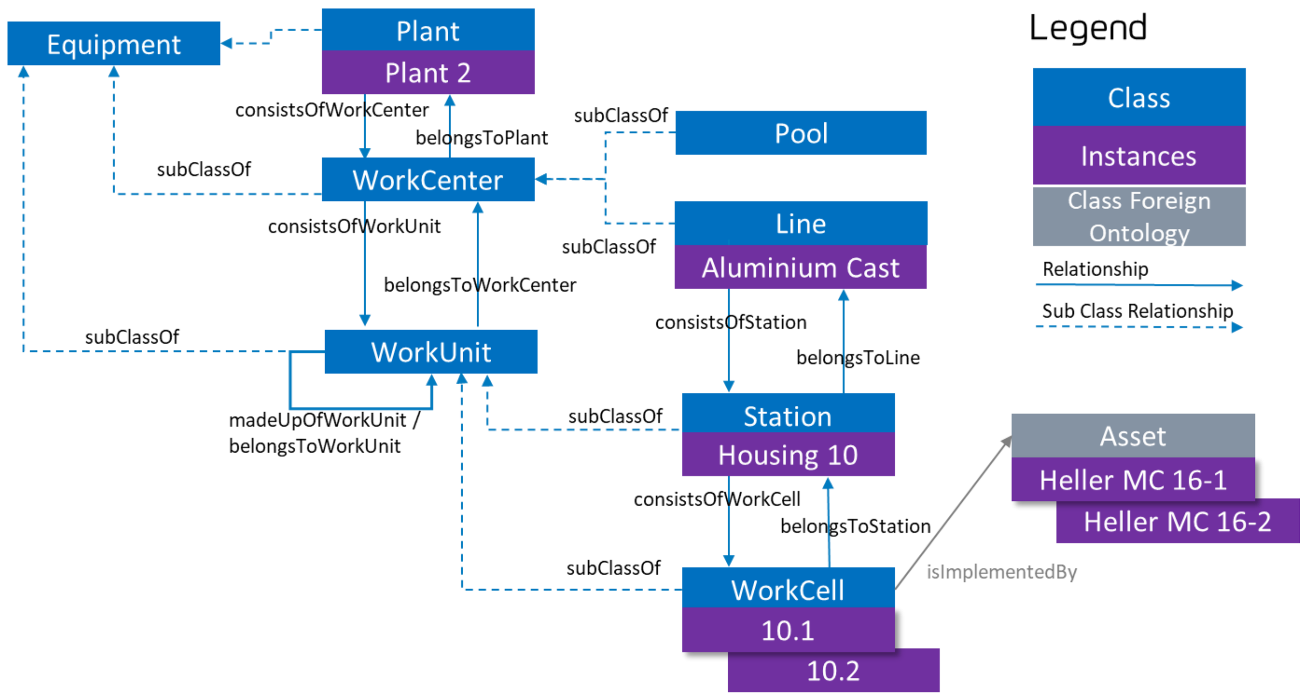


Figure 12: I4.0 Core Information Model - Equipment Model Example

5.5 Process Model

Production process segments are the smallest elements of manufacturing activities that are visible to business processes. The process segment model is a hierarchical model in which multiple levels of abstraction of manufacturing processes may be defined because there can be multiple business processes requiring visibility of manufacturing activities. Process segments are independent of any particular product hence describing general manufacturing process steps.

Process segments describe individual elements of a bigger manufacturing process (see Class Process). For example, the SMT Process is a manufacturing process describing the placement and soldering of electronic components on a PCB board. It consists of the process segments *solder paste printing*, *solder paste inspection*, *pick & place*, *reflow heating,* and *optical inspection*.

Each *process segment* can have an associated *Parameter Specification* (see Parameter Model) to define process parameters. These process parameters typically are used to describe process-relevant setting parameters, e.g., the *temperature* up to which the PCB board is heated up in the *reflow heating process segment*. The process ontology modeled in RDF is available in GitHub.

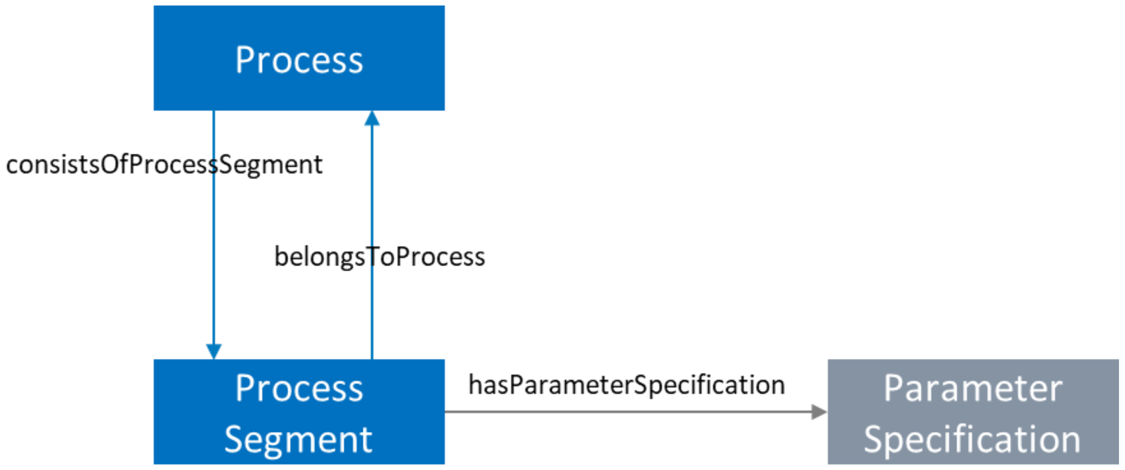


Figure 13: I4.0 Core Information Model – Process Model

Table 3: I4.0 Core Information Model – Process Model – Definition of Classes

Class	**Definition**
Process	Description of a manufacturing process. A manufacturing process may consist of 1..n process segments.
Process Segment	Smallest elements of manufacturing activities that are visible to business processes. Multiple process segments together form a process (manufacturing process). Also called business process segment. Reference: IEC 62264-1 ID:3.1.26

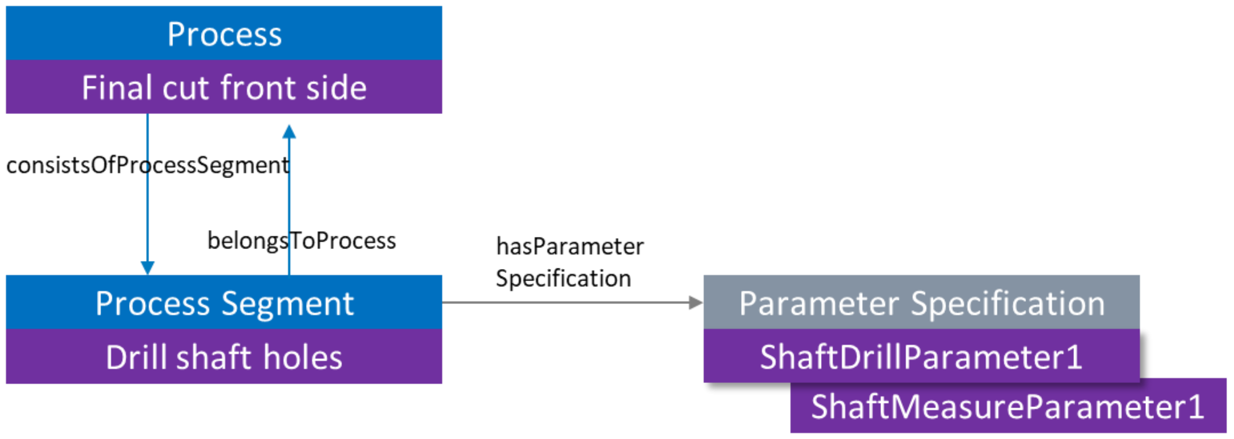


Figure 14: Process Model Example

5.6 Product Model

The product model defines classes required to describe the assembly structure of products (e.g., Bill of Material) as well as product instances. Since the material is defined as self-referencing, it can be recursively de-composed going downwards the hierarchy tree or aggregated into more complex material going upwards the tree. ProcessedPart is introduced to distinguish the hierarchical definition of material (the way the complex materials can be de-composed) from a specific example/instantiation of material. The gearbox of a specific car model and series is a *material*. The gearbox mounted in my own car is the specific *ProcessedPart*. The product ontology modeled in RDF is available in GitHub.

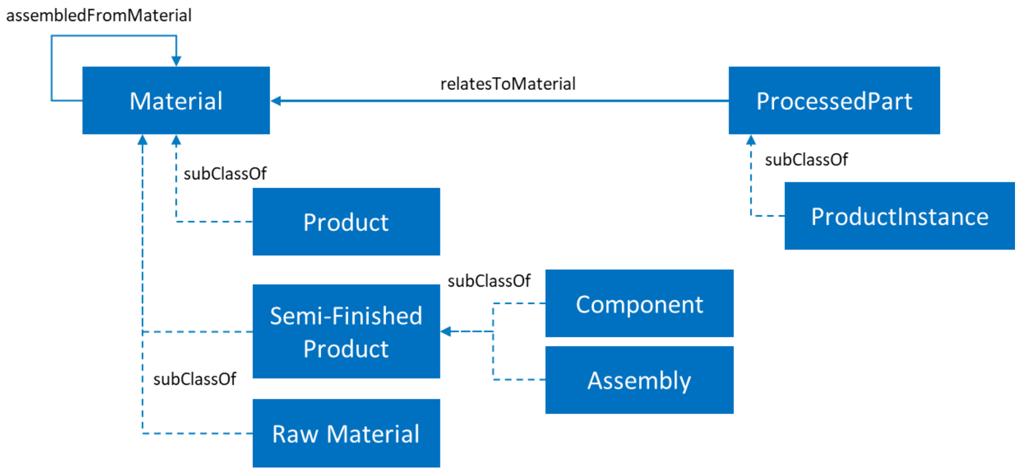


Figure 15: I4.0 Core Information Model – Product Model

Table 4: I4.0 Core Information Model – Product Model – Definition of Classes

Class	**Definition**
Material	Abstraction of a tangible result of some manufacturing or assembly process. A product, semi-finished product, sub-assembly, component, or raw material. Note 1: The term has been purposely selected to create an umbrella for a broad spectrum of possible examples, and to unify the regular case in production chain where outcome of one process/industry (its product), becomes an input of the subsequent process (for example as a component or a raw material). Note 2: Subclasses like product, semi-finished product and raw material allow for more precise and perspective specific classification of materials for the specific context.

Product	Description of a tangible good that is manufactured or assembled and considered ready for being sold to a customer afterwards. Subclass of material. Note 1: Product is a perspective of the one who makes it ready for selling.
Semi-finished Product	Description of a tangible good that is not yet finished. Subclass of material. Note 1: Usually, a semi-finished product is a sub-assembly or a component of another more complex *material* and is not sold to a customer by itself.
Assembly	Description of a tangible good that is the outcome of a previous assembly process. Subclass of material. This includes especially the outcomes of other internal suppliers (e.g., plant A delivering an assembly to plant B). Note 1: It is a perspective of the one who uses it to build a more complex *material*.
Component	Description of a tangible good that is input to an assembly process and is typically purchased from a supplier in a "ready for use" form. Subclass of *material.* Note 1: For example, electronic components, bolts and nuts, gear box for a car OEM. Note 2: It is a perspective of the one who uses it to build a more complex *material*.
Raw Material	Description of a tangible substrate of a specific manufacturing process that is taken in its basic form as input for subsequent manufacturing. It shall not generally be used in the context of tangible input into assembly process. Subclass of *material.* *Note 1: It is a perspective on the one who uses it to manufacture other goods – for example a silicon wafer is raw material for production of semi-conductor electronic devices, where by itself it is a result of complex preparation process, hence a product from perspective of its supplier.*
Processed Part	Description of a unique example/occurrence of a specific *material*. As such it can be an instantiation of any of the *material* subclasses. A processed part refers to *material* it represents via the *"relatesToMaterial"* relationship.
Product Instance	Concrete and unique example/occurrence of a *product* (definition) which is the outcome of manufacturing or assembly. A P*roduct*

	Instance refers to its *product* (definition) via the *"relatesToMaterial"* relationship inherited from *ProcessedPart*.

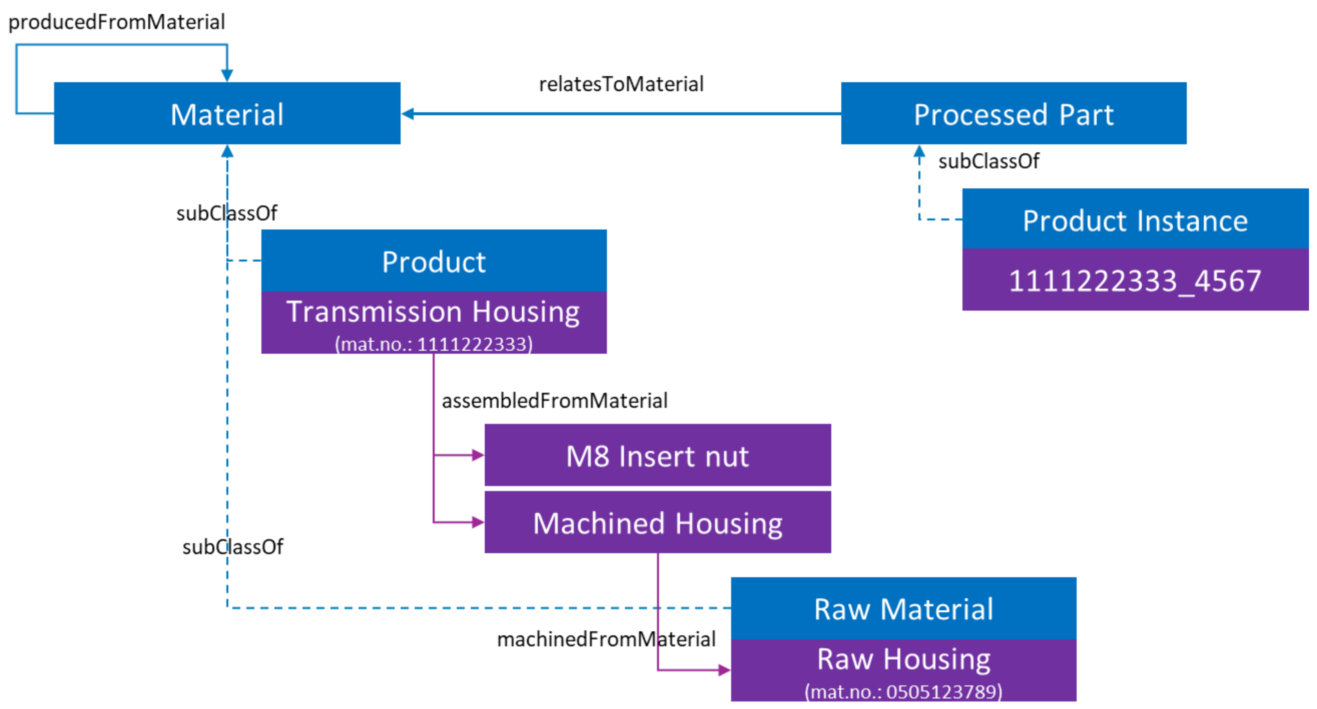


Figure 16: Product Model Example

5.7 Product Segment Model

The Product Segment Model is a product-specific implementation of the process model. Whereas the process model is product-independent, the product segment model is product-dependent. This differentiation is useful if multiple different products are produced based on the same manufacturing process (e.g., SMT Process), but setting parameter values for individual process segments needs to be adapted to the different products.

The product segment model further links the product with the manufacturing process segments as well as the logical equipment (see Equipment Model) where the product will be manufactured. Each product segment implements a process segment. The product segment ontology modeled in RDF is available in GitHub.

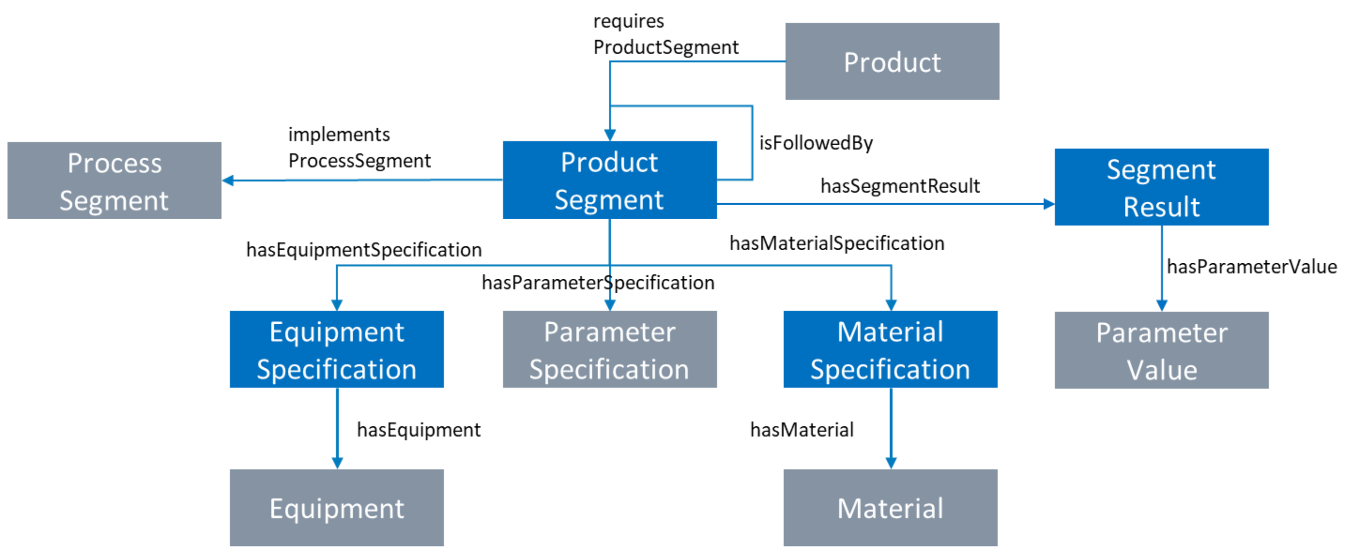


Figure 17: I4.0 Core Information Model – Product Segment Model

Table 5: I4.0 Core Information Model – Product Segment Model

Class	**Definition**
Product Segment	Implementation of a process segment for a concrete product. Thus, a product segment implements a process segment for a concrete product or product variant. A *product segment* identifies personnel, equipment, and material resources required of a process segment to complete a production step for a specific product. Product specific routing information between product segments can be expressed via the *"isFollowedBy"* relationship between product segments. Reference: IEC 62264-1 ID:3.1.29
Segment Result	Outcome of the execution of a product segment for a given product instance or processed part (see product model). A segment result refers to the parameter values of result parameters defined by the parameter specification (see Parameter Model for details).
Equipment Specification	Specification of the required equipment (e.g., stations, work cells) that are required to execute the product segment.

Material Specification	Specifies the material required for executing the product segment. The material specification can refer to the input material and output material of the product segment.
Parameter Specification	Specifies the setting parameters and/or result parameters for a specific product segment (see Parameter Model for details).

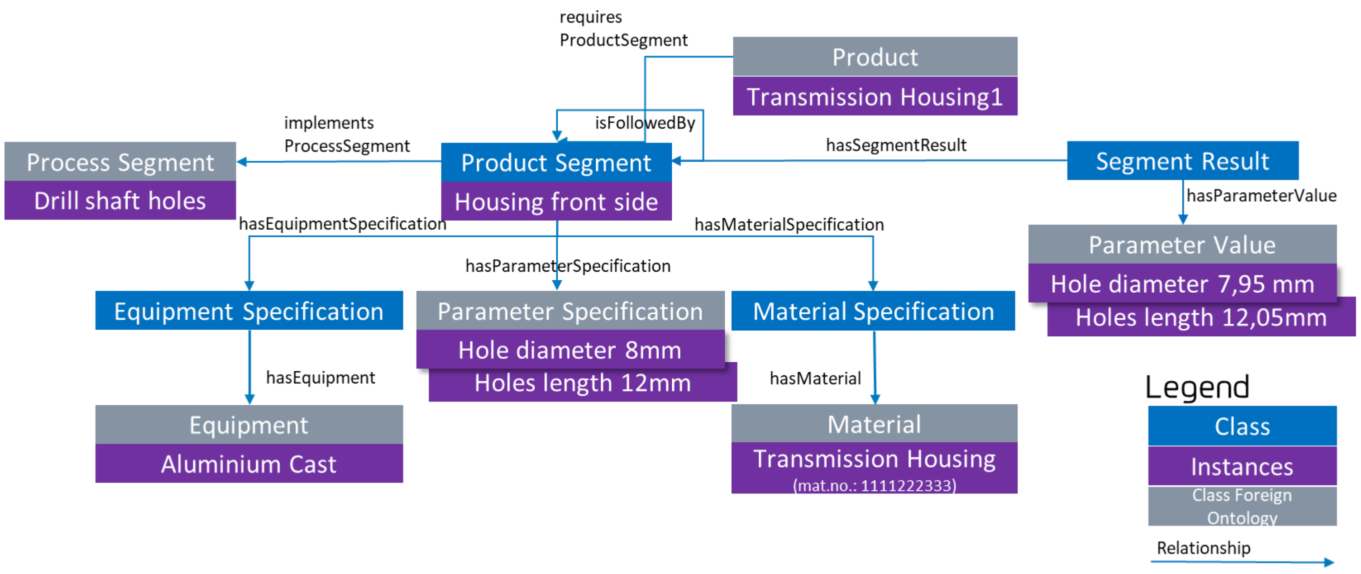


Figure 18: Product Segment Model Example

A Product (e.g., transmission housing) requires several steps to be completed in the context of a finished (or machined) part. A product segment helps to organize the different completion stages of this particular product. An individual segment may be specified with the help of parameters. The actual product segment implementation is done with the help of a process segment linked via the “implementsProcessSegment” relationship. This process segment occurs in this example as a drilling process at the front side of the transmission housing. The process will create a result represented by the segment result class. Those are to be compared to the product segment parameter specification. Tolerances represent additional parameters. The product segment consumes material inside a piece of equipment. Further model extensions are discussed in the fields of energy consumption, waste creation, and recovery (see Annex for details).

5.8 Parameter Model

The Parameter Model is used to describe parameter specifications, parameters, and parameter values. Parameters can be further classified into setting parameters or result parameters. For example, setting parameters are used to specify the setpoint values for, e.g., a manufacturing process or a machine. This might include tolerances if necessary. Result parameters are used to specify which parameters are measured during the execution of a manufacturing process to evaluate the quality of the produced or assembled part. This might include tolerances if necessary.

Each parameter can have a physical unit describing the physical unit (e.g., degree of Celsius for a temperature or kilonewton for a force, …). As there exist multiple very good physical unit ontologies (e.g., QUDT, QU-REC20, or the BAMM Unit Catalog), we keep the unit definition external to the I4.0 Core Information Model. One can simply import a unit ontology or the BAMM Unit Catalog and link the parameter instances to the appropriate unit instances defined in the unit ontology. The parameter ontology modeled in RDF is available in GitHub.

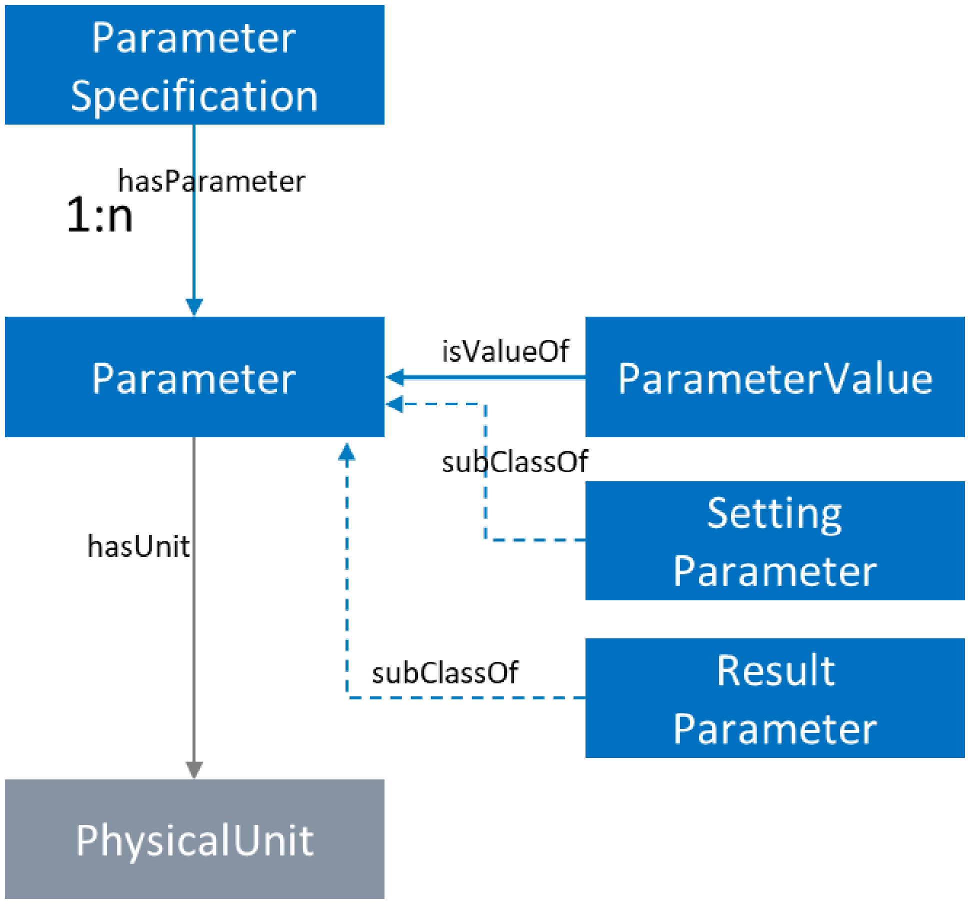


Figure 19: I4.0 Core Information Model – Parameter Model

Table 6: I4.0 Core Information Model – Parameter Model – Definition of Classes

Class	**Definition**

Parameter Specification	Specifies the setting parameters and/or result parameters for a process segment or product segment. A parameter specification consists of 1..n parameters.
Parameter	An individually measured item or property of a process segment or product segment. Parameters belong to a Parameter Specification.
Setting Parameter	Description of input / setting parameters of a given process or product segment. Subclass of Parameter. Note 1: Setting parameters can also contain tolerance values.
Result Parameter	Description of the result parameters of a given process or product segment. Result parameters can also contain tolerance values. Subclass of Parameter
Parameter Value	Concrete value of a parameter. Note 1: For example, the parameter value of the parameter "Temperature" might be "18.5".
Physical Unit	Description of the physical unit of a parameter. Note 1: For example, the parameter temperature has a physical unit of "Degree Celsius". Consider references to external ontologies e.g. QUDT, …

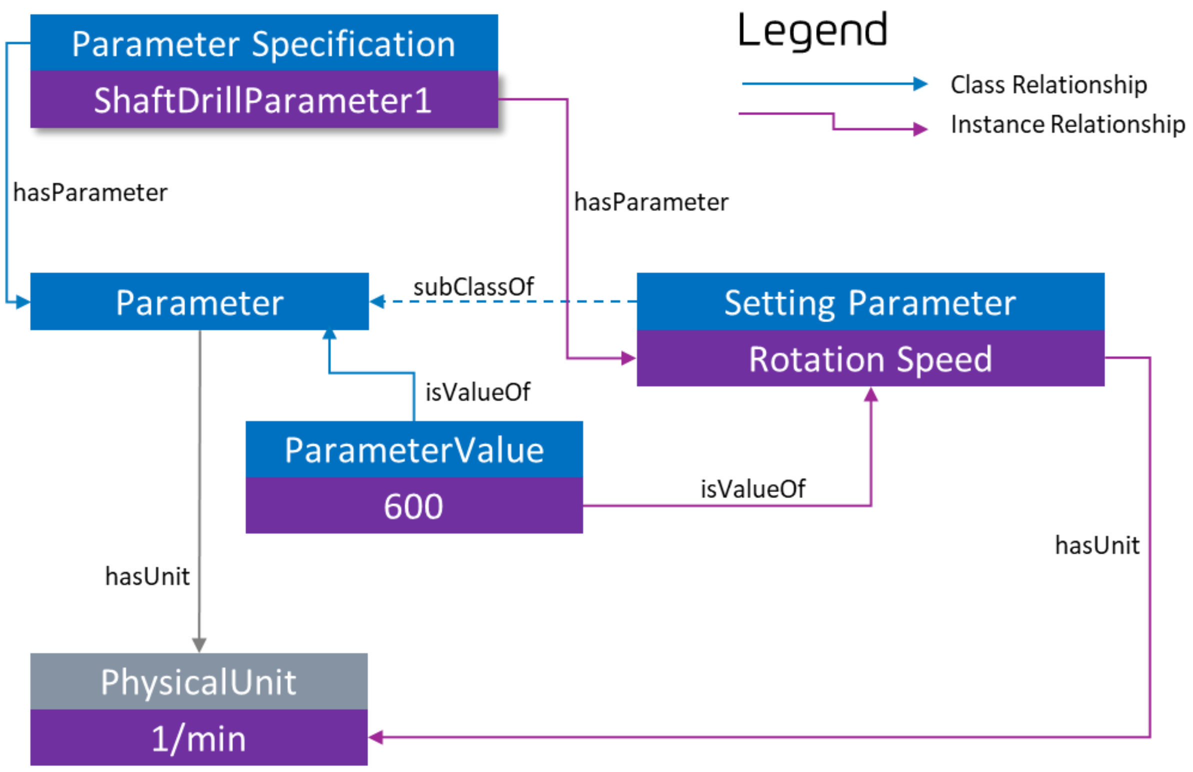


Figure 20: Parameter Model Example

6 Semantic Data Integration

Semantic data integration describes the process of aligning data from at least two or multiple heterogeneous data sources with a defined semantic information model (ontology). Two general flavors of semantic data integration can be differentiated:

- *Materialization*: storage of transformed data in terms of instances of the ontologies in the knowledge graph (at the transformation time)
- *Virtual:* transformed data is generated only at query time, no copy of data), using query rewriting (e.g., SPARQL queries are rewritten as SQL queries and executed on the original data source in case this the data source is relational database and data can be accessed via SQL)

In this whitepaper, we describe in detail the materialization-based flavor of semantic data integration.

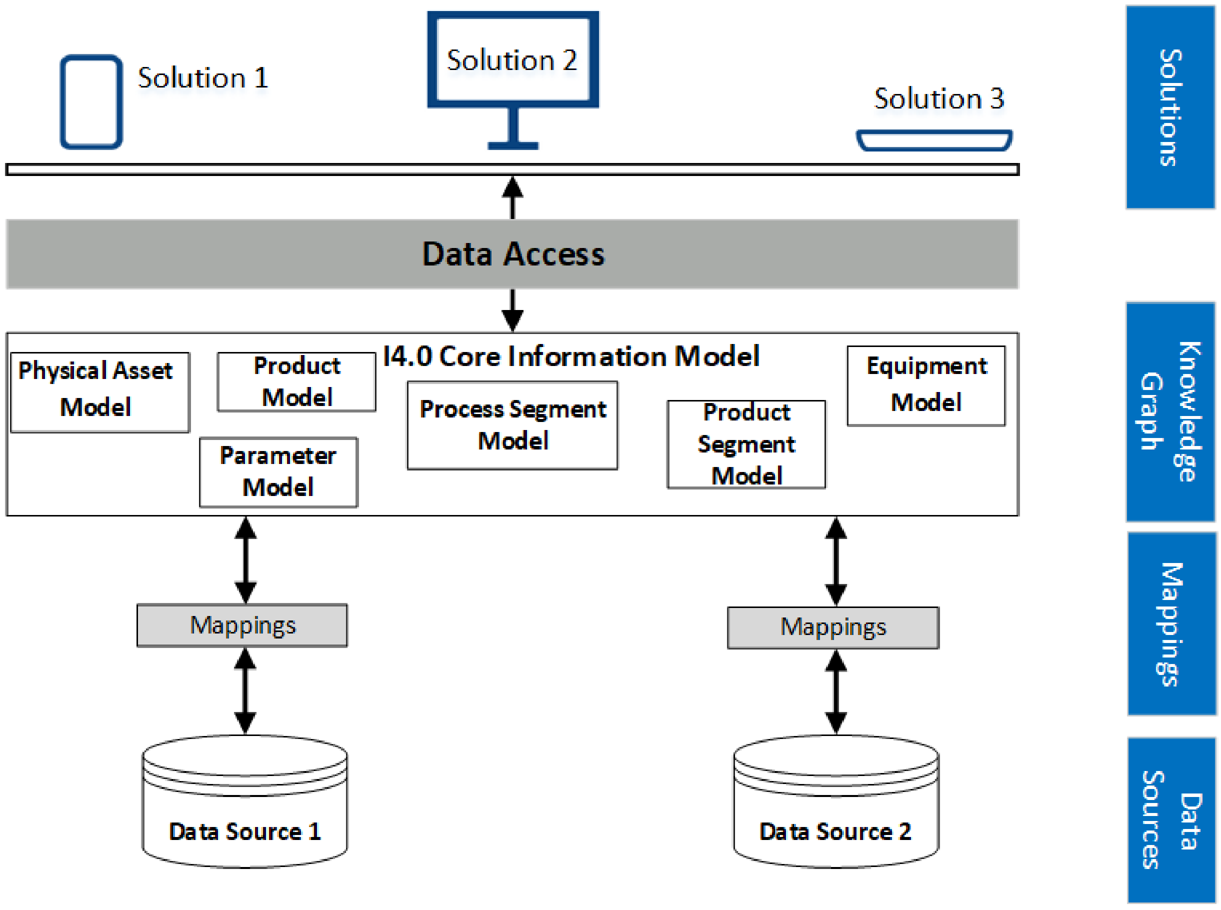
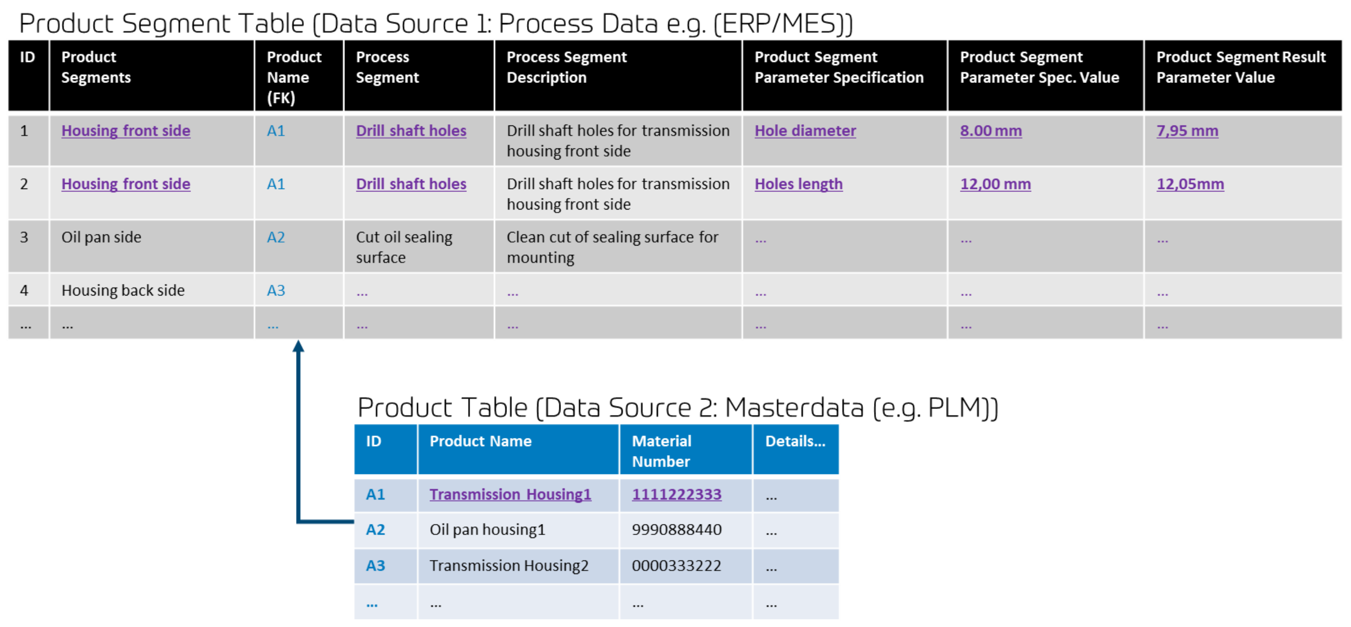


Figure 21: Semantic Data Integration Approach



Product Segment Table (Data Source 1: Process Data e.g. (ERP/MES))

ID	Product Segments	Product Name (FK)	Process Segment	Process Segment Description	Product Segment Parameter Specification	Product Segment Parameter Spec. Value	Product Segment Result Parameter Value
1	Housing front side	A1	Drill shaft holes	Drill shaft holes for transmission housing front side	Hole diameter	8.00 mm	7,95 mm
2	Housing front side	A1	Drill shaft holes	Drill shaft holes for transmission housing front side	Holes length	12,00 mm	12,05mm
3	Oil pan side	A2	Cut oil sealing surface	Clean cut of sealing surface for mounting	...	...	...
4	Housing back side	A3	...	...	...	...	...
...	...	...	...	...	...	...	...

Product Table (Data Source 2: Masterdata (e.g. PLM))

ID	Product Name	Material Number	Details...
A1	Transmission Housing1	1111222333	...
A2	Oil pan housing1	9990888440	...
A3	Transmission Housing2	0000333222	...
...	...	...	...

Figure 22: Exemplary Data Sources for our Production Line Example

In the following, we discuss the architecture for the Semantic Data Integration using the proposed semantic information models. The architecture for Semantic Data Integration comprises four different phases, i.e., 1. Phase: Description of Data Sources, 2. Phase: Creation or Reuse of Semantic Information Model, 3. Phase: Mapping Data to the Semantic Information Model, and 4. Phase: accessing the data.

6.1 Phase 1: Description of Data Sources

In this phase, the data sources have to be described. Particularly, the data fields belonging to the data sources demand to be understood with their proper meaning.

Figure 22 depicts two exemplary data sources for our production line example containing a product segment table (e.g., from an MES system) and a product table (e.g., from a PLM system). For this whitepaper, we have simplified the data sources to a great extent to have an example that is still understandable. However, in reality, the data sources can be much more complex.

6.2 Phase 2: Creation or Reuse of Semantic Information Model

In this phase, a semantic information model is created, or an existing one is reused as a target for integrating data from the data sources. The purpose of the semantic information model is to describe the semantic meaning of the data, its attributes, and relationships to other data elements, which is often lacking in the data sources.

In our case, we use the I4.0 Core Information Model as our target model for integrating heterogeneous manufacturing data sources (e.g., ERP data and MES data).

6.3 Phase 3: Mapping Data to the Semantic Information Model

In this phase, the data from the sources is mapped to the semantic information model. This mapping process is implemented using different mapping languages (e.g., SPARQL or R2RML). The mapping describes how elements of the data sources are mapped to instances of the defined classes in the semantic information model. Usually, the mappings are executed automatically in order to transform the data from the data sources into instances of the semantic information model.

After executing the mappings, the data initially available in the sources is now available as instances of the classes defined in the semantic information model. The combination of a semantic information model (ontology) and the data instances is also referred to as a knowledge graph.

During mapping, the data can also be semantically enriched by adding further metadata or transformed according to the rules defined by the mappings. This enables very powerful operations to be performed on the raw data before the data instances in the graph are created, ranging from data cleaning, disambiguation, decoding, and transformation up to the calculation of derived values.

The following table outlines the mappings for data from the production line example.

Class in I4.0 Core Model	**Table / Column**
Material (Material Model)	Product Table – Product Name Product Table – Material Number
Product Segment (Product Segment Model)	Product Segment Table – Product Segments

6.4 Phase 4: Accessing the data

In this phase, the data previously integrated into a knowledge graph can be accessed using queries. Semantic data integration tools typically offer access to the knowledge graph using, e.g., the SPARQL query language or OpenCypher. Similarly, one can build REST APIs over the knowledge graph, which implement an HTTP interface to access the data. Some platforms also offer direct access to the knowledge graph using Python or other programming languages.

The SPARQL query depicted next describes a straightforward example of retrieving all machine instances that implement the process segment “drill shaft holes”. For simplicity, we have omitted the prefixes of the properties. With these types of queries, the data can be accessed in a uniform and semantic way.

```
SELECT ?machine
WHERE {
?product_segment implementsProcessSegment ?process_segment ;
                          hasEquipmentSpecification   ?equipment_specification .
?process_segment processSegmentName ?process_segment_name .

?equipment_specification hasEquipment ?line .
?line consistOfStation ?station .
?station consistOfWorkCell ?work_cell.
?work_cell isImplementedBy ?machine .
FILTER (?process_segment_name = “drill shaft holes”)
}
```

Alternatively, one can also access the data in the knowledge graph using relational query languages (e.g., SQL). However, this requires converting the semantic information model back to a relational table-based model. This, of course, has the disadvantage of losing the power of the graph-based approach.

7 Relationships to Other Models

7.1 Comparison of I4.0 Core Information Model with Digital Twins and Asset Administration Shell

The main purpose of the I4.0 Core Information Model is to provide a standardized data model for the description of entities, their attributes, and relationships that are relevant in a manufacturing context.

The BAMM Aspect Meta Model is a meta-model for describing so-called aspect models (information junks) that can describe data exchanged via the Digital Twin of a specific entity (e.g., a machine on the shopfloor, equipment, a product, or a manufacturing process).

The following models, the I4.0 Core Information Model, the BAMM Aspect Meta Model, and the BAMM aspect models are based on the semantic modeling language RDF.

The BAMM Aspect Meta Model can be instantiated to create concrete BAMM aspect models (for example, the movement aspect of an AGV). Our I4.0 Core Information Model can as well be instantiated to create concrete models for a specific purpose (e.g., see the example of the production line in Section 5).

However, they serve different purposes. The I4.0 Core Information Model is particularly suitable if multiple heterogeneous data sources need to be integrated into a manufacturing context and for analytical use cases also requiring historical data. In contrast, the BAMM Aspect Meta Model and BAMM aspect models are suitable if the task is to semantically describe the data exchanged via a Digital Twin or a data interface. Furthermore, digital twins and the BAMM Aspect Meta Model are very well suited if one is interested in the real-time update of data/values of a physical asset and the control of the physical asset via its digital twin.

The Asset Administration Shell is the implementation of a digital twin of Industrie 4.0. Hence, the above-mentioned points for comparing the I4.0 Core Information Model and the Digital Twin also apply to the Asset Administration Shell.

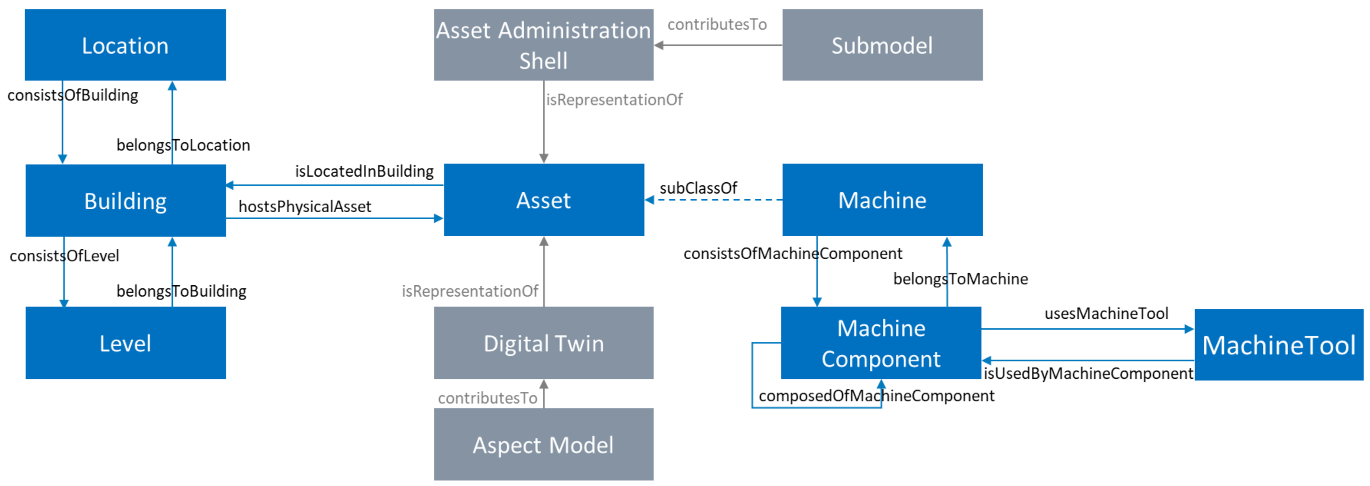


Figure 23: Relations of Physical Asset Model to AAS and BAMM

Both models – the I4.0 Core Information Model and the BAMM Aspect Meta Model and BAMM aspect models are based on RDF. They can also be combined and used together. For example, it is possible to generate an aspect model from an ontology (e.g., the I4.0 Core Information Model) and vice versa, thus enabling the use of both models in combination in a use case.

There are several ways to combine both technologies. For example, data can first be integrated and mapped to the I4.0 Core Information Model to create a knowledge graph. Then aspect models can be used to define Digital Twins on top of the integrated data in the knowledge graph to serve the data to applications. Likewise, it is also possible to first define aspect models and digital twins and later integrate the data provided by the aspect implementations of the digital twins into a knowledge graph.

7.2 Comparison to OPC UA Information Model

Open Platform Communication Unified Architecture (OPC UA) attempts to address the problem that "systems [in all industrial domains] are intended to exchange information and to use command and control for industrial processes. OPC UA defines a common infrastructure model to facilitate this information exchange OPC UA specifies the following:

- The information model to represent structure, behavior, and semantics.
- The message model to interact between applications.
- The communication model to transfer the data between endpoints.
- The conformance model to guarantee interoperability between systems."[3]

For the purpose above, OPC UA defines two communication architecture models, Client-Server (for coupled communication) and Publisher-Subscriber (with an option to introduce middleware Broker for decoupled exchange of information), paying attention to security, interoperability, and portability aspects of the constructed infrastructure.

Proposing an infrastructure model for the service and application layer, OPC UA defines a blueprint and set of concepts that can be reused by implementations in the specific industrial domain and specialized within companion specifications. Defined infrastructure targets are independent of the transport layer used for information exchange.

One of the powerful concepts in OPC UA is the representation of information sources/sinks and relations between them inside an OPC UA *Server*[4] as *AddressSpace* (a multi-layer graph, where *Nodes* are vertices and *Relations* between *Nodes* are arcs of the graph), with respective standardized *Services* that allow OPC UA *Applications* discover and analyze the semantics of these graphs. The challenge for a user of this concept is the reach set of (proto-)types necessary to handle this infrastructure, which is an initial learning challenge. Moreover, being an infrastructure specification written with an object-oriented mindset, OPC UA allows for independent implementations in the specific programming languages used by derived deployments.

OPC UA also provides an Information Model, which, regardless of the similarity of name, should not be confused with the Core Information Model discussed in this whitepaper. According to OPC UA, "The

[3] from OPC 1000-1, Chapter 5, Release 1.04

[4] Terms in "italics" like *Node, AddressSpace, Service*, and so on should be understood as defined by respective OPC UA specifications

Information Model describes standardized *Nodes* of a *Server's Address Space*. These *Nodes* are standardized types as well as standardized instances used for diagnostics or as entry points to server-specific *Nodes*. Thus, the Information Model defines *AddressSpace* of an empty OPC UA *Server*."[5]

From the perspective of infrastructure specification, OPC UA Information Model is a blueprint for *Server* implementation and a common baseline that allows OPC UA Applications to conduct the Server discovery. However, it still does not describe the aspects of the specific industrial application or domain, which are defined during the server implementation stage.

Relating the contents of this white paper to OPC UA, the following shall be underlined:

- Regardless correct usage of the term Information Model in both contexts, the meanings are not exchangeable. OPC UA Information Model refers to the organization of *AddressSpace* of OPC UA *Server*, while Core Information Model in this whitepaper refers to modeling of production means and processes on Level 3 and 4 of Industrial Pyramid and refines the IEC 62264 concepts. Therefore, the two differ in their intention and applicability.
- Infrastructure defined by OPC UA specifications can be adopted to provide language for the creation of physical assets or equipment models as defined in this whitepaper
- OPC UA conformant deployment of Level 1 and 2 can be mapped/transformed to the model described in this whitepaper or can be used to build OPC UA conformant extensions toward Core Information Model. The effort needed and usability of such an approach shall be the driving factors here.
- Regarding production process-specific aspects captured by this whitepaper's Process, Product, and Parameter models, they seem to escape the focus of OPC UA specifications as these cannot be obviously reflected by the architecture focused on communication and interoperability in the control domain. OPC UA's attempt regarding this type of modeling is not known to the authors of this whitepaper.

8 Use Cases

The I4.0 Core Information Model can be used for a large number of different use cases. In order to get a better understanding, we briefly give a high-level description of the different usages of the Core Information Model.

[5] from OPC 10000-5, Chapter 1, Revision 1.05.00

8.1 Process Quality Monitoring

The I4.0 Core Information model provides the means to describe process chains, including process result parameters and their tolerances. This provides the basis for process quality monitoring use cases. In this case, the data coming from manufacturing machines or a Manufacturing Execution System (MES) is mapped to the I4.0 Core Information Model using the Semantic Data Integration approach described in Section 6 to obtain a knowledge graph holding the process quality result data for a large number of manufactured product instances.

Analytics on the process quality (e.g., in the form of dashboards) or more advanced AI algorithms can then be developed on top of the knowledge graph without connecting directly to the data sources.

8.2 Production Planning

The I4.0 Core Information model can also be used for production planning. The process model provides a way to describe the general manufacturing processes, whereas the product segment model provides ways to describe concrete implementations of the process segments for concrete products. Important setting parameters for tailoring a manufacturing process for a concrete product can be precisely described and modeled, including the mapping of processes to concrete production lines and concrete machines.

8.3 Manufacturing Traceability

A high-level requirement for any manufacturing company is to provide traceability, i.e., to trace material and assembly information down to the assembled raw materials. The I4.0 Core Information Model provides many model entities needed to implement manufacturing traceability, starting from the material model defining a recursive model of the assembled materials down to the product segment model defining entities for the description of the result of manufacturing processes and individual parts assembly.

8.4 Asset Inventory

The I4.0 Core Information Model provides with the physical asset model a good way to implement asset inventory use cases in which one needs to know which assets are located in which buildings and production lines. An asset inventory system can be implemented easily on top of a knowledge graph which instantiates the I4.0 Core Information model.

9 What’s Next?

This white paper is the starting point for the definition of a common and standardized I4.0 Core Information Model in the Open Manufacturing Platform. We have pointed out several use cases which can significantly benefit from the I4.0 Core Information Model defined in this white paper.

As a next step, we will implement the I4.0 Core Information Model as a semantic model and publish this information model as Open Source in OMP.

10 Glossary

(Core) Information Model

An information model in software engineering and data modeling is an abstract, formally represented conceptual data model expressing concepts, attributes and relationships, constraints, rules, and operations to specify data semantics for a chosen domain of discourse. An information model thus defines the blueprint or schema for data. Data or data instances are usually not part of an information model.
Source: Wikipedia

Ontology

In computer science and information science, an **ontology** encompasses a representation, formal naming, and definition of the meaning of the categories, properties, and relations between the concepts, data, and entities that substantiate one, many, or all domains of discourse (e.g., the manufacturing domain). Ontologies can be used for the standardization of entities and terms across a number of heterogeneous data sources. Ontologies are represented using formal languages such as RDF, RDFS, and OWL.
Source: Wikipedia
For example, an ontology describing organizations has the concepts “Employee” and “Department” and the relationship “employeeWorksInDepartment” linking “Employee” with “Department”.

Knowledge Graph

A knowledge graph formally represents semantics by describing concepts, attributes, and relationships (ontology) in combination with instances of these entities (instance data). A knowledge graph can be constructed from heterogenous data sources by applying the process of semantic data integration.

Data Preparation

This is the act of manipulating (or pre-processing) raw data (which may come from disparate data sources) into a form that can readily and accurately be analyzed, e.g., for business purposes. Data preparation is the first step in data analytics and can comprise many different sub-tasks such as data ingestion or loading, data cleaning, data transformation, data fusion, and data integration.
Source: Wikipedia

Data Cleaning

This is the process of detecting and correcting (or removing) corrupt or inaccurate records from a record set, table, or database and refers to identifying incomplete, incorrect, inaccurate, or irrelevant parts of the data and then replacing, modifying, or deleting the dirty or coarse data.
Source: Wikipedia

Data Transformation

This is the process of converting data from one format or structure into another format or structure. It is a fundamental aspect of most data integration tasks.
Source: Wikipedia

Semantics

In computer science, the term *semantics* refers to the meaning of language constructs or strings, as opposed to their form (syntax). Source: Wikipedia
For example, the string "Apple" can heave two different semantic meanings. One meaning of "Apple" is the fruit, and another meaning of the string "Apple" is the company. Ontologies can be used to express the knowledge in a domain by using semantic concepts and interrelationships.

Semantic Data Integration

Semantic data integration integrates raw data from heterogeneous data sources (e.g., a database, a CSV file, a JSON file, a data lake, or an IT system) to a semantic data model (ontology). Semantic data integration involves mapping, transformation, harmonization, and alignment of data and data formats to the ontology. The outcome of the whole process of semantically integrating data and aligning it with the semantic data model (ontology) results in a knowledge graph of a specific domain (see Knowledge Graph for further definition).

Semantic Context

Semantic context is defined as an abstraction layer where some form of semantic context – usually business terminology – is provided to the data abstraction.

Data Source

This is any kind of source of data. Relational/NoSQL/Graph Databases, IT systems, REST APIs, CSV files, JSON files, and XML files can all be data sources.

Digital Twin
A digital twin is a virtual representation of real-world entities and processes, synchronized at a specified frequency and fidelity.
Source: https://github.com/digitaltwinconsortium/dtc-glossary/blob/main/glossary.md

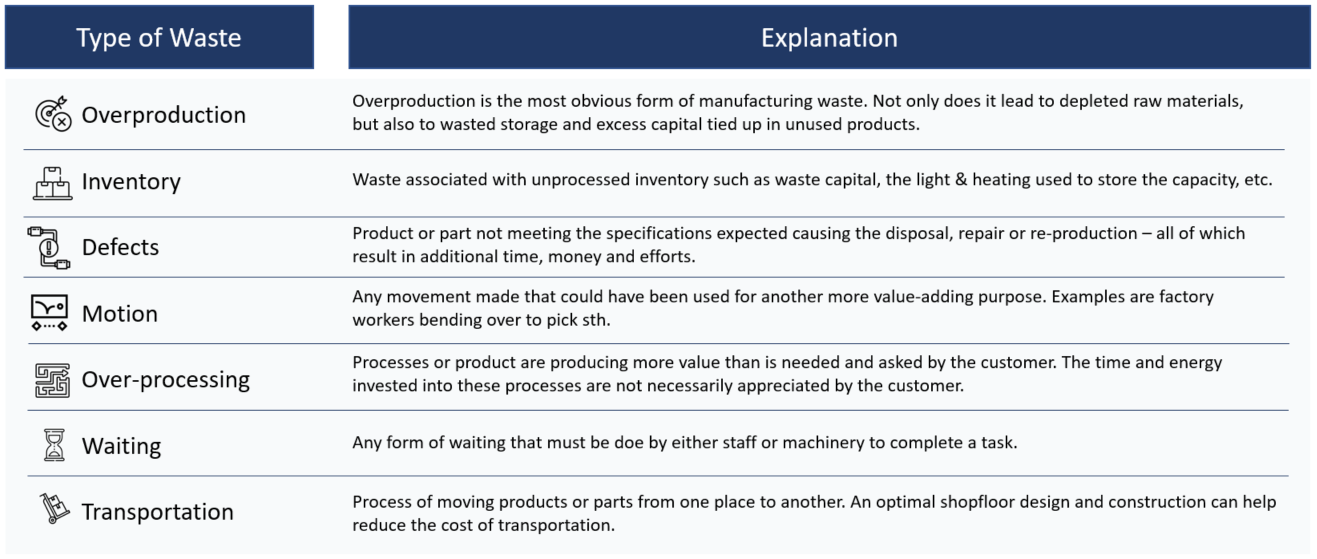
Asset Administration Shell
Digital representation of an asset.
Note 1: Each administration shell can contain one or multiple part models.
Note 2: Administration shell is a synonym for asset administration shell (AAS).
Note 3: The administration shell exists within one phase or across different phases of the lifecycle.
Source: Industrie 4.0 – Begriffe/Terms, VDI Statusreport Industrie 4.0 (April 2021)

11 Appendix

This appendix lists possible extensions of the I4.0 Core Information Model.

11.1 Waste Taxonomy

In production, a well-accepted concept of waste can be found in the Lean (Six) Sigma Practice that goes back to the Toyota Production System. What is expressed by the Japanese term ‘muda’ is a categorization of typical waste types in production. These categories are instrumental in making factory workers and managers aware of waste and defining measures to reduce as much of it as possible. Based on this concept, the following waste categorization and explanation can be summarized from a production perspective.



Type of Waste	Explanation
Overproduction	Overproduction is the most obvious form of manufacturing waste. Not only does it lead to depleted raw materials, but also to wasted storage and excess capital tied up in unused products.
Inventory	Waste associated with unprocessed inventory such as waste capital, the light & heating used to store the capacity, etc.
Defects	Product or part not meeting the specifications expected causing the disposal, repair or re-production – all of which result in additional time, money and efforts.
Motion	Any movement made that could have been used for another more value-adding purpose. Examples are factory workers bending over to pick sth.
Over-processing	Processes or product are producing more value than is needed and asked by the customer. The time and energy invested into these processes are not necessarily appreciated by the customer.
Waiting	Any form of waiting that must be doe by either staff or machinery to complete a task.
Transportation	Process of moving products or parts from one place to another. An optimal shopfloor design and construction can help reduce the cost of transportation.

Figure 24: Types of Waste – Own illustration based on PTC - The 7 Wastes of Lean Production (adapted from *https://www.ptc.com/en/blogs/iiot/7-wastes-of-lean-production)*

Figure 25 illustrates the first ideas to tackle the ontology elements about waste. Waste may be differentiated in energy, material, or emissions. Likewise, energy or scrap materials may be recovered through special equipment or processes.

Waste may occur in different contexts. One might consider it in the context of finished goods. A final product and waste are created at the same time. It may occur in a particular process segment. Or is aggregated on plant level.

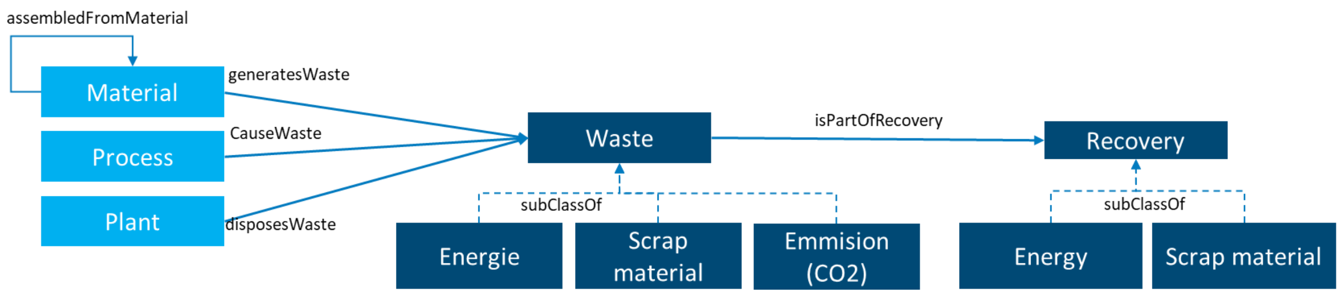


Figure 25: Exemplary draft for a waste taxonomy and linking to the I4.0 Core Information Model

11.2 Product Characteristic Model

Digitalization changes how products are being designed and manufactured and even what is considered a product. Data packages, software, algorithms, and other non-tangible goods affect manufacturers as well as classical software companies.

Time to reflect on the semantics of a product information model:

- What is a product or a Service?
- What sub-structures emerge?
- What relationships exist?

11.3 Organizational & Financial Model

The equipment and asset model often reflect technical views on the manufacturing equipment. Yet, the financial and company internal organization of assets should not be disregarded. The Core I4.0 Information model could be extended with two new dimensions.

First, the financial model will provide an ontology for financial objects (e.g., cost elements). In a second step, it will be linked to existing model elements. Additionally, the organizational model may provide semantics about organizational objects (e.g., departments).

These enrichments cover non-technical aspects of *Industry* 4.0.

11.4 Missing IEC Element Extensions

The IEC 62264 Standard is a powerful source for deriving an Industry 4.0 Core Information Model. However, the standard applies to many industries. Some manufacturing and Industry 4.0 specific objects, relationships, and context is missing.

Example elements might tackle:

- Intra-logistic
- Working models
- human work steps, ...

As long as a manufacturing context is identified, the model may grow in all directions.