Towards a Comprehensive Methodology for Modelling Submodels in the Industry 4.0 Asset Administration Shell

Cornelis Bouter
Data Science
TNO
Soesterberg, The Netherlands
cornelis.bouter@tno.nl

Monireh Pourjafarian
Innovative Factory Systems
DFKI
Kaiserslautern, Germany
monireh.pourjafarian@dfki.de

Leon Simar
VDL Industrial Modules
Helmond, The Netherlands
l.simar@vdlindustrialmodules.nl

Robert Wilterdink
Monitoring & Control Services
TNO
The Hague, The Netherlands
robert.wilterdink@tno.nl

***Abstract*—The Industry 4.0 Asset Administration Shell is designed as a digital communication interface that exposes information of various aspects of an asset in Submodels, which are functional blocks. However, a methodology that identifies the set of Submodels required for a specific use case is lacking. This work therefore presents two results: 1) the methodology we developed to identify the Submodels applicable in a use case and 2) an example application involving immaterial process assets. Our approach complements existing procedures by beginning with a functional description rather than an established standard and by creating a potentially interconnected set of Submodels rather than a singleton. The outcomes were verified by domain experts and through integration of the corresponding AASs.**

***Index Terms*—Industry 4.0, asset administration shell, submodel, methodology, immaterial asset, entity-relationship diagram, knowledge representation**

I. Introduction

The Asset Administration Shell (AAS), developed under the Industry 4.0 umbrella, aims to provide a common digital representation of each factory asset. It plays a major role in the goals of enabling data sharing between value chain partners, standardizing data security, and providing technology-neutral semantic standards [1]. The role of the AAS specifically is to provide communication among assets within a single factory and cross-company, and to cover the complete asset life-cycle from requirements until decommissioning [1].

Industry 4.0 adopts a broad definition of what constitutes an asset: anything of value in the factory such as machines, raw materials, services, and human personnel. Moreover, immaterial assets such as processes and plans are explicitly included as well [2]. The asset information is stored on the AAS in standardized Submodels, which are data models each pertaining to a specific functionality. Several Submodels have been published for machine AASs [3], [4] which have become de facto standards.

We however encountered a lack of established Submodels for immaterial and non-machine assets. This is problematic for the standardization and interoperability efforts because Submodels provide the information abstraction required for data exchange. The Submodel scarcity therefore leads to the development of different models for similar functionalities.

Moreover, a methodology to identify the necessary Submodels for a use case has not been established. The existing Submodel development procedure [5] describes how to translate a single established standard into an Industry 4.0-compliant Submodel, but a use case will likely contain multiple assets with various functionalities. Also, the existing procedure results in a single Submodel, whereas a use case may require several Submodels to properly express the domain information. A methodology that identifies the set of Submodels required for a use case is therefore lacking.

In this work we present our research towards a Submodel development methodology that starts at a use case description and concludes with deployment as a measure of validation. The intermediate steps cover both technology-independent conceptual modelling as well as deliberate design decisions to comply with Industry 4.0 concepts. As an example we apply the methodology on a Dutch high-mix low-volume factory. The goal was to get insight into the factory shop floor performance and to simultaneously make the Enterprise Resource Planner (ERP) data available for other process improvement projects in the H2020 DIMOFAC project[1]. The shop floor information contains various types of assets instead of only machine assets: processes and orders, produced products and raw materials, and services and human personnel. This broad scale sets an appropriate test case for the methodology.

This work will commence with the state-of-the-art Industry 4.0 literature in section 2, followed in section 3 by the basics of the AAS. We present our methodology in section 4. The example application of the methodology on our use case is described in section 5. This work concludes with a discussion of the AAS deployment in the use case.

This paper is supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No 870092, project DIMOFAC (Digital Intelligent MOdular FACtories).

[1]https://dimofac.eu/

II. State of the art

The normative AAS documentation [2] has been published by Platform Industry 4.0 in co-operation with ZVEI, the German Electrical and Electronic Manufacturers' Association. The document formally describes the contents of an AAS and its components. The communication of the AAS to the outside world is defined with a technology-independent API specification in the follow-up document [6]. Part 3 covers the Industry 4.0 communication language among AASs, but remains unpublished as of July 2021.

Additional non-normative material published by the Platform Industry 4.0 describes how to use the AAS from the perspective of different roles [7]. In [5] it is shown how the AAS data should be structured into several Submodels. How to use the AAS elements to construct a composite AAS is described in [8], [32]. Additional practical considerations and example Submodels are given in [9]. The philosophy of enabling semantics by linking each concept to a global identifier is argued for by the Platform Industry 4.0 in [10] and independently in [11], [12]. The terms and definitions behind the Industry 4.0 architecture are also clarified in [13], elaborating on the difference between AAS and Digital Twin.

The AAS is technology-independent by design. In [2] a mapping is specified to XML, JSON, RDF, OPC-UA, and Automation ML, with each having a different purpose. The XML and JSON serialization are intended for technical communication, RDF to enable semantic technologies, AutomationML [14] to share asset information during the engineering phase, and OPC-UA [15] for sharing live data. Additional remarks on the RDF and OPC-UA serialization are given by [16] and [17], [18] respectively.

The actual data is stored in Submodels, each covering a separate asset functionality. Example machine Submodels have been provided for Drilling, MES connection, Energy Efficiency, and Documentation functionality [19]. The Nameplate Submodel [4] and Technical Data Submodel [3] are widely used for machine AASs. An example Submodel for PLCOpen [20] has been published outside of the Platform Industry 4.0 initiative. The process of standardizing machine skills and capabilities has been described in [10], [21], although a Submodel implementation is not provided. Additionally, for several potential Submodels the relevant established standard has been identified (figure 2).

Some early research has been done towards Submodels for non-machine assets, such as a MES connection [19] or order and supply management [9]. In [22] an AAS is developed for an operator wearing a smart jacket, although no Submodel is presented. There is to the best of our knowledge no publication on representing in a Submodel immaterial assets, in particular work orders or process steps.

Currently, only a few real use cases have made use of AAS technology. The research has focused on normative documentation, machine Submodels, or small-scale deployment. For example, [23] employs the AAS in a plug-and-produce scenario, where three robotic arms take turns performing a simple task. Another work [24] describes a demonstration with an AAS for PLC data. In [22] a operator jacket is developed and deployed using AAS technology.

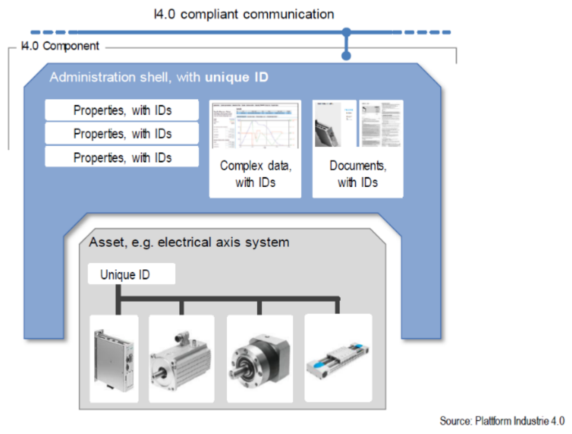

I4.0 compliant communication
I4.0 Component
Administration shell, with unique ID
Properties, with IDs
Properties, with IDs
Properties, with IDs
Complex data, with IDs
Documents, with IDs
Asset, e.g. electrical axis system
Unique ID
Source: Plattform Industrie 4.0

Fig. 1: Structure of an AAS [2]

Several AAS tools are currently under development. The AASX Package Explorer[2] is a tool for constructing AASs and Submodels. It additionally has eClass import functionality, options for Submodel reuse, and functionality to expose an AAS via OPC UA or REST. The associated aasx-server[3] provides functionality to expose multiple AASs following the XML serialization. BaSyX[4] is intended as an Industry 4.0 Standard Development Kit available for Java, C#, and C++, supporting the major Industry 4.0 communication protocols [25]. Additionally, PyI4.0AAS[5] is a Python implementation of [26]. An AAS digital integration platform is also being developed in the H2020 DIMOFAC project.

III. Basics of the AAS

Figure 1 shows the basic AAS structure: Each asset (grey box) has a unique Administration Shell (blue box) that serves as an interface to the asset information. The AAS functions as a digital communication shell that exposes the asset data to other Industry 4.0-comliant components. The actual data is stored into various Submodels, each covering a separate asset functionality.

The AAS equivalents of attributes are called SubmodelElements [2]. For example, a Property contains a literal value of a specified datatype; a ReferenceElement contains a reference to a Referable (e.g., an AAS, Submodel, or SubmodelElement); a Document contains an actual file or a link; and a SubmodelElementCollection contains a set of SubmodelElements.

The AAS requires SubmodelElements and the Submodel itself to be identifiable through a globally unique identifier [2] for standardization and semantic integration. An International Registration Data Identifier (IRDI) can be used to refer to

[2]https://github.com/admin-shell-io/aasx-package-explorer
[3]https://github.com/admin-shell-io/aasx-server
[4]https://projects.eclipse.org/projects/technology.basyx
[5]https://pypi.org/project/pyi40aas/

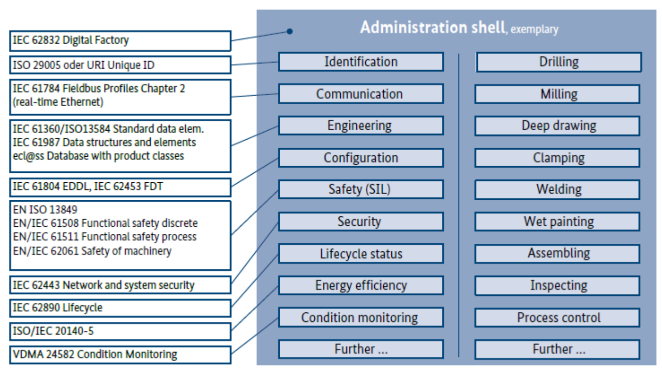

Administration shell, exemplary
IEC 62832 Digital Factory
ISO 29005 oder URI Unique ID
IEC 61784 Fieldbus Profiles Chapter 2 (real-time Ethernet)
IEC 61360/ISO13584 Standard data elem.
IEC 61987 Data structures and elements
ecl@ss Database with product classes
IEC 61804 EDDL, IEC 62453 FDT
EN ISO 13849
EN/IEC 61508 Functional safety discrete
EN/IEC 61511 Functional safety process
EN/IEC 62061 Safety of machinery
IEC 62443 Network and system security
IEC 62890 Lifecycle
ISO/IEC 20140-5
VDMA 24582 Condition Monitoring
Identification
Communication
Engineering
Configuration
Safety (SIL)
Security
Lifecycle status
Energy efficiency
Condition monitoring
Further ...
Drilling
Milling
Deep drawing
Clamping
Welding
Wet painting
Assembling
Inspecting
Process control
Further ...

Fig. 2: Potential Administration Shell Submodels [8]

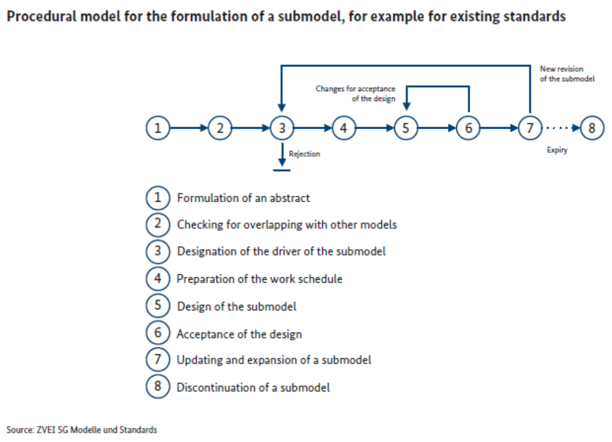

Procedural model for the formulation of a submodel, for example for existing standards
Changes for acceptance of the design
New revision of the submodel
Rejection
Expiry
1 Formulation of an abstract
2 Checking for overlapping with other models
3 Designation of the driver of the submodel
4 Preparation of the work schedule
5 Design of the submodel
6 Acceptance of the design
7 Updating and expansion of a submodel
8 Discontinuation of a submodel
Source: ZVEI SG Modelle und Standards

Fig. 3: Procedure for Submodel development [5]

standardized dictionaries such as eClass or another relevant standards [27], as is done in the example Submodels [3], [4], [19]. An Internationalized Resource Identifier (IRI) can refer to a (semantic) web resource [10].

An important design principle of the AAS is that Submodels should be separated by functionality [5] such that all assets that share a specific function have the same Submodel implemented on their AAS. Interoperability is then achieved by defining for each aspect a standard Submodel that should be used cross-company [10]. For example, the *Asset Umgebung* (= Asset Environment) Submodel [9] contains the relevant information of an asset in different life cycle phases. Such a Submodel is applicable independent of asset type and use case.

A Submodel development procedure is proposed in [5] (Figure 3). It is primarily intended to develop an Industry 4.0 compliant Submodel from existing standards.

IV. Methodology

The methodology we present aims to identify and produce the Industry 4.0-components necessary for a specific AAS application scenario. Different scenarios involve different functionalities and assets. The appropriate Submodels and AASs have to be identified from the extensive Industry 4.0 scope for each use case separately.

For example, the integration of an Automated Guided Vehicle (AGV) into the production process requires driving capabilities and location information from the vehicle itself. Additionally, planning data should be exposed for the AGV to read its next task. Use case analysis would identify Submodels for capabilities (driving), location, and planning. The machines, AGVs, processes, and parts would be identified as assets requiring an AAS.

Predictive maintenance is another potential AAS application scenario. The Submodels identified for the previous example would be irrelevant in this case. Instead, data concerning the lifecycle and condition monitoring would be compared with engineering and configuration data. The methodology would therefore identify the Lifecycle Status, Condition Monitoring, Engineering, and Configuration Submodels from figure 2 to be required.

We take a high-level perspective and describe the methodology functionally: It takes as input a use case description together with a goal the AAS deployment is intended to obtain. The methodology produces three results:

1) A technology-independent conceptual model describing the domain
2) The design of the necessary Submodels
3) A preliminary deployment to validate the Submodels

These three results are produced by the three distinct phases (figure 4) of which the methodology consists, respectively. The knowledge representation phase collects a vocabulary of relevant terms and relates the selected concepts with each other into a conceptual model. In the Industry 4.0-compliance phase the conceptual model elements are mapped to AAS components. The third phase concerns evaluation of the Submodels through a validation and a verification step. The former checks whether the Submodels fit the use case; the latter whether the Submodels correctly support the functional requirements.

The methodology contains two evaluation steps: 1) the conceptual model is validated in the final step of the knowledge engineering phase, and 2) the Industry 4.0 components are reviewed in the evaluation phase. If either of those evaluation steps produces a negative result, we start another iteration of the methodology starting at either the knowledge representation phase or Industry 4.0-compliancy phase depending on the severity of the issues. Alternatively, we can adopt an Agile way of working by selecting only a subset of requirements for each iteration. A positive evaluation can then lead to another iteration including additional requirements.

A. *Knowledge engineering phase*

The Knowledge Engineering phase consists of three steps: requirements analysis, development of the conceptual model, and an evaluation step. The requirements analysis is explicitly part of the knowledge engineering phase because the methodology does not assume the existence of an established standard. A conceptual model can therefore only be derived after defining the requirements.

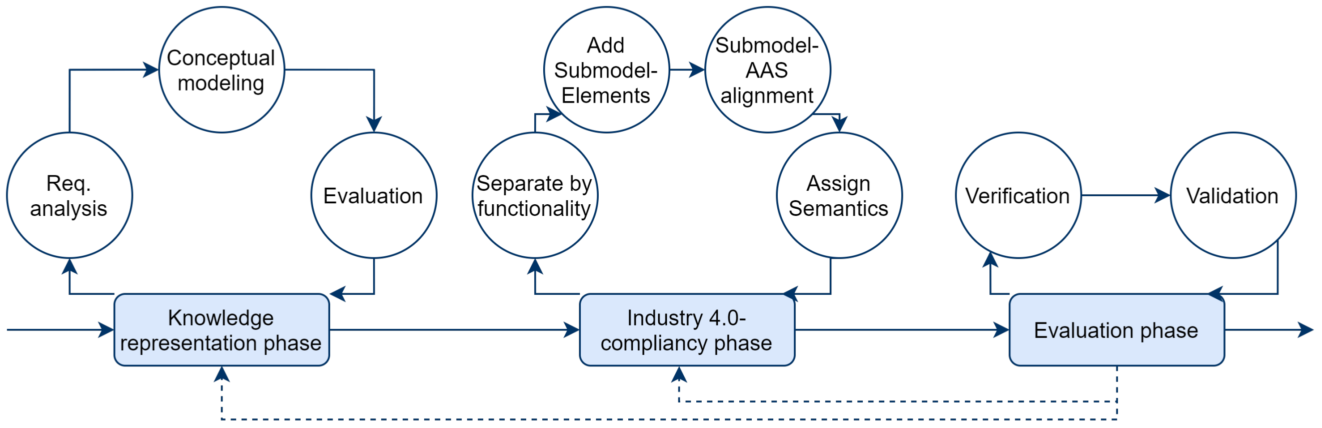


Fig. 4: Overview of the methodology

1) Requirement analysis: This step should produce a set of functional requirements in natural language from a use case description. Various conceptual model formalizations, such as UML models, Entity-Relationship (ER) diagrams, and ontologies, offer differing ways to describe the functional requirements. The standard practice for describing functional requirements in ontologies is a list of competency questions [28], [29]. These natural language questions describe the queries that the model needs to answer in the use case. Requirements for UML models and ER diagrams usually are given as a list of statements the model should conform to [30].

2) Conceptual model: The requirements elicited from the use case are then used to develop a conceptual model. The choice of formalization may depend on the use case circumstances. ER diagrams may ease communication with database engineers who are already familiar with that paradigm. Also, the distinction between entities and relations provides suggestions in the Industry 4.0-compliance phase, as will be detailed in the section IV-B. UML models may provide easier communication with software engineers. They also force the specification of datatypes, which is required for Submodels. The Web Ontology Language (OWL) is the W3C standard for ontology formalization. It has the use of IRIs as globally unique identifiers in common with the AAS. Another advantage is that ontologies support a modular design similar to the AAS [10].

3) Intermediate review: The knowledge engineering phase produces a conceptual model as its result. This model should be reviewed by a domain expert through a visualization method that is available for the three modelling paradigms we consider. The conceptual model can also be used to compare the use case with other use cases.

B. Industry 4.0-compliancy phase

In the second step the technology-independent conceptual model is adapted to Industry 4.0 components. These introduce additional non-functional constraints on the model. The result of this phase is a design for a set of Submodels based on the conceptual model.

The Submodels identified during this phase may already have been implemented. It is recommended to reuse existing standards and Submodels to benefit from previous efforts and to maximize interoperability. A custom set of Submodels has to be implemented following the identified design, when no suitable Submodels can be found. The approach in [5] can be followed in the case of a relevant established standard, except if the use case requires a holistic view with multiple Submodels and Assets.

1) Separation by functionality: In the Basics of the AAS section we described that Submodels are separated by functionality to improve reusability of a single Submodel among several AASs. The Industry 4.0 technology follows the SOLID software engineering principles [31] in this regard:

1) A single Submodel contains the data concerning a single functionality, following the single-responsibility principle.
2) Multiple Submodels exposing small subsets of the data is preferred over a single Submodel exposing all the data. This follows the interface segregation principle.
3) The AAS also follows the dependency inversion principle, which states that abstractions instead of concretions should be used, by distinguishing between Submodel templates and instances.

Therefore, if a concept contains multiple attributes unrelated to each other, it most likely contains multiple functionalities each requiring a dedicated Submodel.

It is important to consider that the Submodels produced via this methodology are intended to be reused in future use cases. This makes sure that the AAS can continuously rely on previous efforts to benefit further standardization. For example, energy information may be identified via our methodology as a necessary Submodel that has not been designed and implemented yet. If the Submodel is then implemented, it may be disseminated such that it becomes a standard that is used in future use cases involving energy consumption information.

Formalizing the conceptual model as a UML class diagram benefits this step, because the UML metamodel formalizes various relationships between objects. For instance, an aggregation or composition suggests the placement of all Submodels in a single AAS because of the 'part-whole' relationship that is modelled.

2) Submodel elements: The conceptual model relations then have to be mapped to AAS SubmodelElements. We identify two types of relations independent of conceptual model formalization:

1) Attributes contain a literal value or another value that is not part of the model itself, such as a document. These attributes form an endpoint of the modelling effort.
2) Relationships connect a model concept to another model concept.

The AAS provides several options to represent model attributes, as the AAS is geared towards implementation instead of representation. We list the most commonly used elements:

1) The **Property**, which can optionally be a multilanguage property, simply contains a value with a specified datatype. A built-in type of the XML Schema Definition 1.1 is required[6].
2) A **SubmodelElementCollection** is a container for a collection of SubmodelElements including itself.
3) A Submodel can refer to data via the **File** or the **Blob** SubmodelElement, containing an (online) file or binary data, respectively.

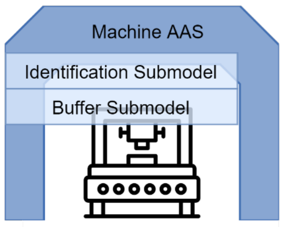
The AAS provides a **ReferenceElement** and a **RelationshipElement** to model relationships between model concepts. The former specifies a reference, while the latter specifies a relationship between a subject and an object. The semantics of the element define the type of relationship. These elements are used to relate AASs to each as specified in section IV-B3.

3) Asset and AAS alignment: In this phase we decide for each model concept whether it is a factory asset that merits its own AAS or whether it is a collection of attributes pertaining to an existing asset. In the former case a dedicated AAS is created for each instance of that concept. In the latter case only a Submodel is created and implemented on the AAS of another model concept. Two examples can illustrate the difference.

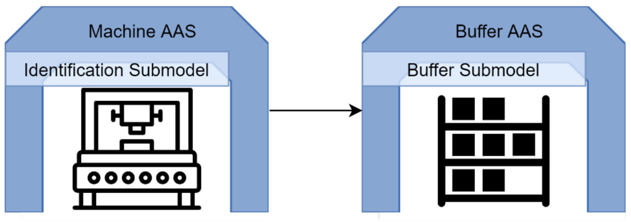
Consider a CNC machine containing an internal material buffer. The Submodel containing buffer information may be implemented on the AAS already dedicated to the CNC machine (figure 5a). The other option (figure 5b) is to create a separate AAS for both the machine and the buffer. These can be connected as specified in [32] using ReferenceElements.

Alternatively, consider a Submodel containing data concerning a sales order. When a customer places an order, the sales department requests a work order to fulfill the demand. The Submodel therefore relates a customer AAS and a work order AAS to each other, while additionally adding sales information such as price and deadlines. Figure 6 shows that we can implement the sales order Submodel on either AAS.

[6]https://www.w3.org/TR/xmlschema11-1/

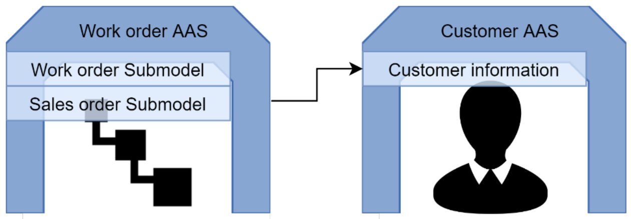


(a) Implementing two Submodels on the same asset

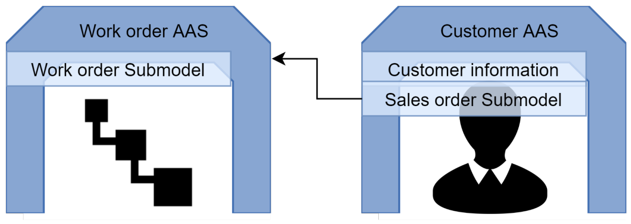


(b) Separate AASs for each Submodel

Fig. 5: Options for to model a machine buffer Submodel



(a) Implemented on the work order AAS



(b) Implemented on the customer AAS

Fig. 6: Options to model a sales order Submodel

Formalizing the conceptual model as an ER diagram gives a benefit in this step, since the distinction between entity and relationships provides a rule-of-thumb: Entities can generally be regarded as assets with their attributes contained in one or more Submodels. We identify three different ways of aligning the basic ER diagram structure (figure 7) with the Industry 4.0 paradigm.

1) In figure 8a a dedicated AAS is created for the relationship, so the concept modelled as an individual AAS is regarded as providing value on its own. If the relationship only provides value in the context of another concept, it should be modelled on another asset as in the

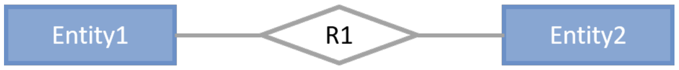
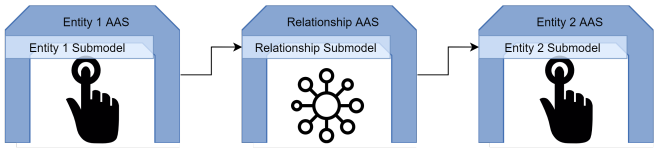
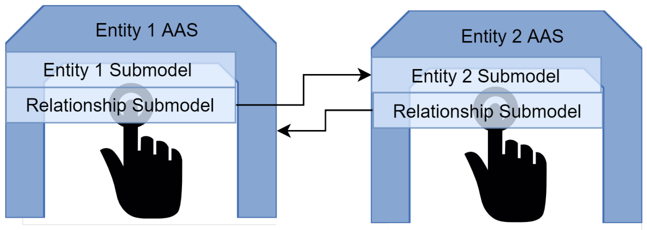


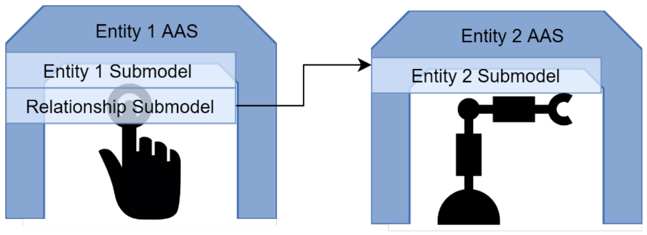
Fig. 7: Generic view of two entities and a relationship



(a) Dedicated AAS for the relationship



(b) Relationship Submodel on either AAS



(c) Relationship Submodel on stable AAS

Fig. 8: Entity and relationship modelling options

following two options.

2) In Figure 8b we consider the relationship to be part of another asset. Implementing the Submodel on either AAS is considered equally viable, if the connection to both AASs is stable and constant.
3) In the third option the relationship also is only relevant in connection with another AAS. If the connection to either one of the AASs is flexible and may often change during deployment, the Submodel should be implemented on the stable AAS (figure 8c).

4) Semantics: The final step of the Industry 4.0-compliance phase is to assign semantics to each Submodel and SubmodelElement, such that each element has a globally understood meaning.

A newly developed Submodel should receive an IRI using the namespace of the developing organization. The semanticIDs of SubmodelElements, however, should be drawn from external established standards such as an IEC/ISO standard or eClass to enhance data interoperability. Only when there is no standard available that covers a specific attribute, a custom IRI can be created following the format prescribed in [2].

Formalizing the conceptual model as an ontology benefits this step, because identifying concepts through IRIs is already required. Ontologies engineering methodologies also strongly support reuse of existing models and standards.

C. Evaluation phase

We conclude the methodology with evaluating the Submodels that were designed and implemented in the previous phase. The Submodels are verified by checking that each functional requirement is covered. Validation takes place through deployment in the intended use case.

The machine or factory data may be available in various formats. The AAS technology closely follows OPC UA development to feed the AAS with machine data. However, market penetration of OPC UA remains low, despite it being recognized as an Industry 4.0 standard [33]. Factory data, especially ERP or MRP data, may also be available in a relational database system (RDBMS) that needs to be exposed through an AAS via queries (e.g., SQL). Alternatively, the data may be available as OWL triples needing SPARQL queries [34] or another graph protocol. Verification is considered complete when for each functional requirement it is confirmed that the information can be exposed through the AAS.

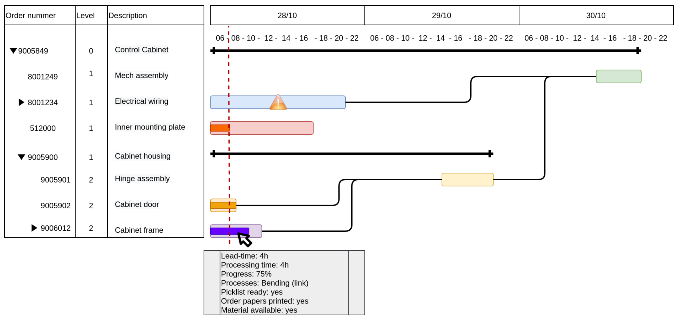
Additionally we need to validate that the Submodels developed indeed solve the problems stated in the use case. This step simultaneously validates whether the functional requirements exhaustively described the use case. Otherwise, the whole methodology should receive another iteration. A positive validation would constitute an actual deployment of the AASs in the manner of a demo or proof of concept. Some shortcomings may only be identified at this step.

V. Example Methodology Application

We applied the methodology in a use case for a Dutch high-mix low-volume manufacturing company. The selling point of the company is to have in-house capabilities to adapt data for both manufacturing engineering and production processes. This means the company can evaluate the customer designs to adapt it to production capabilities, which leads to a high percentage of first-production among orders. The combination of high-mix low-volume and the large percentage of first-production orders requires a high level of flexibility.

The factory ERP system stores all production planning related information. Each night a Material Requirements Planning (MRP) algorithm calculates the optimal production schedule based on the current requirements such as delivery deadlines. The computed schedule is on a per day basis (i.e., without specific start times) and additionally assigns each work order a priority. During the day, the work orders are executed following the priority list. To increase flexibility and operating efficiency, the company wants to connect the factory floor real-time with the ERP system. The promise of the AAS providing a flexible framework for information and functions has led to this use case.

In this scenario flexibility is the ability to decide on the next work order when a task is completed. This choice should be based on work order dependencies and real-time statuses, such



Order nummer	Level	Description
▼9005849	0	Control Cabinet
8001249	1	Mech assembly
▶ 8001234	1	Electrical wiring
512000	1	Inner mounting plate
▼ 9005900	1	Cabinet housing
9005901	2	Hinge assembly
9005902	2	Cabinet door
▶ 9006012	2	Cabinet frame

Lead-time: 4h
Processing time: 4h
Progress: 75%
Processes: Bending (link)
Picklist ready: yes
Order papers printed: yes
Material available: yes

Fig. 9: Work order planning Gantt chart

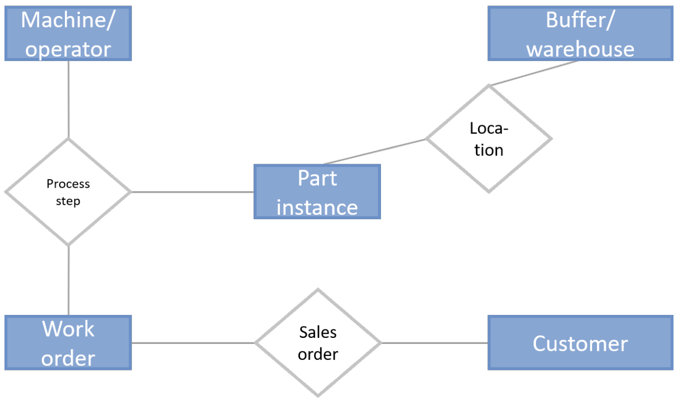


Fig. 10: ER diagram (excluding attribute names for clarity)

as the warehouse inventory, and changes in planning priorities. The high-mix low-volume nature requires a lot of change-overs. Since change-overs do not directly add value to the product, it is essential to only start on work orders that can be successfully finished. Using real-time shopfloor status the operator knows which tasks can be started. Additionally the planning department can use the same information to react immediately when issues arise, e.g. when a defective part is identified, and mitigate the situation. This combination ensures that no time is wasted on unnecessary change-overs, which in turn leads to a higher throughput and more stable lead-time.

A. *Knowledge representation phase*

We examined the scenario, conducted interviews with the factory planners, and inspected the ERP data model. We also inspected the components of the Gantt chart (figure 9) describing the factory processes. This examination led to the development of a vocabulary of terms and a set of competency questions (table I).

The **Work order** is the collection of all process steps required to produce a part.

A **Process step** is a low-level during which an action is performed by a machine on some part. It corresponds with the process concept from the PPR model [10], [35].

A **Resource** is the generalization of the object to perform a work order. It can be either a **machine** or a **human operator**.

The **Part instance** is a representation of the physical part that is present in the factory. Three types have been identified: raw material, sub-assembly, and finished product.

The **Customer** is the party either demanding or supplying some product or part.

A **Buffer** is a generalized storage space within the factory. Three types of storage have been identified: machine buffer, non-machine temporary buffer, and warehouse.

We collected the terms into an ER diagram (figure 10) following the competency questions. We based the Process step relationship on requirements 1, 6, and 8. Following requirement 1 a process step is part of a work order. Requirement 6 specifies that a process step should have two related parts: the consumed part and the produced part. Following requirement 8 the process step is related to the machine it is performed on. We similarly analyzed the other requirements.

The model was evaluated in collaboration with an employee of the Dutch factory where the use case is situated who works on a daily basis with their database. The model was also reviewed by an internal SCSN and modelling expert.

B. *Industry 4.0-compliancy phase*

In the Methodology section we described the four phases the Industry 4.0-compliance phase consists of: separating Submodels by functionality, assigning SubmodelElement types to the attributes and properties, aligning the assets with the Submodels, and assigning semantics. Since we developed the conceptual model as an ER diagram, we use the rule-of-thumb that each entity can be mapped to an AAS.

We identified one relationship from the model that covers multiple functionalities: the work order. Requirement 7 states that the work order should contain its work orders and process step dependencies. The other work order requirements state that the model should contain status and planning information. Therefore we divide the work order functionality in two separate Submodels:

- **Work order** Submodel covering requirements two to six, inclusive. The general work order data regarding planning and status is contained in this Submodel.
- The **Work order dependency tree** Submodel covers requirement 7, containing both the process step as well as work order dependencies. This Submodel can therefore be seen as a Manufacturing Bill of Materials.

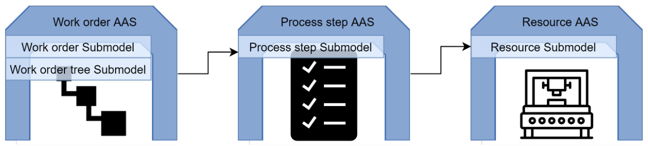
The second part of the Industry 4.0-compliance phase is to store each attribute in the appropriate SubmodelElement type. Each simple attribute was mapped to a Submodel Property. In some cases we collected several Properties collectively containing the address information into a SubmodelElement-Collection.

A deliberate design choice was required on the deployment of Submodels representing the relationships of the ER-diagram: process step, sales order, and location.

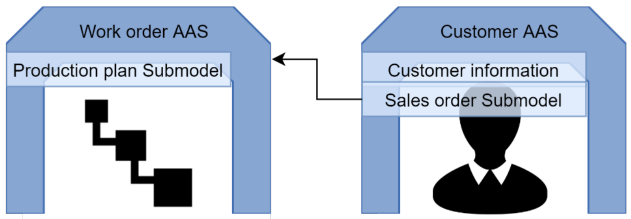
In the use case analysis we described that optimization happens at the **process step** level. Therefore, the process step is an (immaterial) object of value, which merits a dedicated AAS following the Industry 4.0 design philosophy (figure 11a).

Req	Question	Response datatype
Req 1	Which process steps does a work order consist of?	Process step reference list
Req 2	What is the status of a work order?	Status code
Req 3	What is the work order identification number?	Integer
Req 4	What is the starting time of the first process step of a work order?	DateTime object
Req 5	What is the end time of the last process step in a work order?	DateTime object
Req 6	Which parts are produced and consumed by a process step?	(consumed part reference, produced part reference)
Req 7	Which other work orders and process steps is a work order dependent on?	Work order reference list
Req 8	On which machine is a process step scheduled?	Machine reference
Req 9	What are the planned start and end times of a process step?	(DateTime object, DateTime object)
Req 10	Where is a part currently located?	Buffer reference
Req 11	When was a part placed at this location?	DateTime object
Req 12	What is the maximum and current capacity of a buffer?	(Integer, Integer)
Req 13	Which customer ordered for the product or part to be produced?	Customer reference

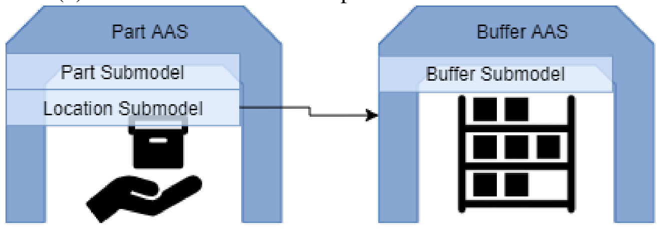
TABLE I: Competency questions



(a) The process step as a separate asset



(b) Sales order Submodel implemented on customer AAS



(c) Location Submodel implemented on part AAS

Fig. 11: Submodel to AAS alignment design choices

A **sales order** has no value by itself as it gains meaning through its relation to a production plan and a customer. The design choice whether to implement the Submodel on the Customer or the Work order AAS can be decided based on the use case (figure 8b). If the focus lies on customer analysis, the Sales order Submodel can be implemented on the Customer AAS, but when the focus is put on the work order to collect all information regarding the order, then it is recommended to model it on the Work order AAS. In our case, we implement the sales order Submodel on the Customer AAS (figure 11b).

The **location** of a part only has meaning in relation to the part and the buffer, so it remains a Submodel on another AAS. The location of a part in a factory is volatile, so the Submodel should be implemented on the stable asset (figure 11c).

We reuse identifiers from the established eClass and IEC CDD vocabularies to give the Submodel semantics. Additionally we reused the message standard of the Dutch Smart Connected Supplier Network[7] project which has developed a model for sharing information regarding invoices, orders, and bills of material. Additionally, it contains references for concepts such as customer and person. The message standard is based on the Universal Business Language (UBL) [36]. We used SCSN identifiers for the properties in the Sales order and the Customer Submodel. In the case where eClass, IEC CDD, and SCSN provided overlapping identifiers, for example on address information, we assigned several semanticIDs to a single SubmodelElement.

We also identified overlap between the Submodels identified by the methodology and already established Industry 4.0 models. The Nameplate and Identification Submodels concern general machine information, so these can be reused for our application. The part, work order, and resource models mirror the components of the PPR (Process, Product, and Resource) model.

C. Evaluation Phase

The methodology identified the following Submodels to be necessary for the examined use case. These were consequently modelled via the AASX Package Explorer tool.

The **Work order** Submodel provides information regarding status and planning.

The **Work order dependency tree** combines work order dependencies and process step dependencies into a single tree structure.

The **Process step** contains detailed production information, such as the time schedule, progress, and both start and end times. The Submodel additionally contains ReferenceElements to the machine that performs the operation and the work order of which the process step is part.

The **Sales order** Submodel details the agreement that led to the production of some product, including references to the involved parties and the production plan.

The **Part instance** Submodel contains basic information about a part, such as a CAD drawing and a BOM file.

[7]https://smart-connected.nl/

The **Customer information** Submodel contains general information regarding a customer, such as contact information and addresses.

The **Location** Submodel connects a part to the buffer where it is currently located.

The **Buffer** Submodel contains the maximum capacity and current capacity of the storage facility.

We verify the Submodels by integrating the AASs with the factory ERP system. We develop a SQL connection extending the AASX-server that exposes the AAS with live ERP system data. This requires specifying for each Submodel the query to be sent to the database. The query result variables are aligned with the elements of the Submodels. A database change event triggers an update of the AAS data.

Validating the Submodel requires showing that the AAS configuration (figure 12) can solve the problems identified in the factory use case analysis. In the use case analysis we described several ways of providing real-time insight into the shopfloor. The data exposed via the AASs can be used for any of those. We implement a Gantt chart functionality based directly on Submodel data as an example of an AAS based tool.

VI. Discussion

The methodology we presented complements the existing Submodel development procedure [5] (Figure 3) in several ways. Firstly, the existing procedure is intended for established standards. The specification of the first phase ("Formulation of an abstract") includes intermediate steps on standards to be used as a source and standardization bodies to be consulted. Our procedure requires a larger role for best practices from the domain of knowledge representation and a smaller role for existing standards.

Secondly, the existing procedural model describes the development of a single Submodel, whereas our method considers the use case as a whole, as the problem may not be reducible to a single submodel. The existing procedure can be applied in parallel to develop multiple Submodels, but it does not include a way to identify the set of Submodels to be developed. The existing procedure can therefore be applied when an established standard should be used as the basis for an undeveloped Submodel identified by our methodology.

Another point of discussion that merits further research is the AAS deployment infrastructure. A multitude of AASs is created already in our limited example (figure 12) so the number of AASs may be several times higher in a production environment. Deploying the AAS as effectively a multi-agent system will put more demands on the IT environment supporting the AASs [37], [38].

VII. Conclusion

In this work we have presented a methodology for identifying the set of Administration Shells and Submodels required for a specific Industry 4.0 application scenario. This fills the gap in AAS research of a comprehensive methodology containing both knowledge representation and incorporating AAS technology. We showed the value of our methodology by giving an example application on a high-mix low-volume use case which produced the set of necessary Submodels.

Our additional contribution is the development of Submodels. We created several Submodels for the immaterial assets such as a work order and a process step that are intended for further reuse.

References

[1] "Details of the asset administration shell from idea to implementation," Plattform Industrie 4.0, 2019.

[2] "Details of the asset administration shell - part 1," Plattform Industrie 4.0, ZVEI, 2020, version 3.0.

[3] "Submodel templates of the asset administration shell - Generic frame for technical data for industrial equipment in manufacturing," Plattform Industrie 4.0, ZVEI, 2020, version 1.1.

[4] "Submodel templates of the asset administration shell - ZVEI digital nameplate for industrial equipment," Plattform Industrie 4.0, ZVEI, 2020, version 1.1.

[5] "Structure of the asset administration shell," Plattform Industrie 4.0, ZVEI, 2016.

[6] "Details of the asset administration shell - part 1," Plattform Industrie 4.0, ZVEI, 2019, version 2.0.

[7] "Functional view of the asset administration shell in an Industrie 4.0 system environment," Plattform Industrie 4.0, 2021.

[8] "Relationships between I4.0 components - Composite components and smart production," Plattform Industrie 4.0, ZVEI, 2017.

[9] "Verwaltungsschale in der Praxis," Plattform Industrie 4.0, 2020.

[10] "Describing capabilities of Industrie 4.0 components," Plattform Industrie 4.0, 2020.

[11] V. Jirkovský and M. Obitko, "Enabling semantics within Industry 4.0," in *Industrial Applications of Holonic and Multi-Agent Systems*, V. Mařík, W. Wahlster, T. Strasser, and P. Kadera, Eds. Cham: Springer International Publishing, 2017, pp. 39–52.

[12] M. Obitko and V. Jirkovskỳ, "Big data semantics in Industry 4.0," in *International conference on industrial applications of holonic and multi-agent systems*. Springer, 2015, pp. 217–229.

[13] C. Wagner, J. Grothoff, U. Epple, R. Drath, S. Malakuti, S. Grüner, M. Hoffmeister, and P. Zimermann, "The role of the industry 4.0 asset administration shell and the digital twin during the life cycle of a plant," in *2017 22nd IEEE international conference on emerging technologies and factory automation (ETFA)*. IEEE, 2017, pp. 1–8.

[14] B. Wally, "AutomationML provisioning for MES and ERP - Support for IEC 62262 and B2MMML," AutomationML, 1 2021, version 2.0.0.

[15] J. Fuchs, J. Schmidt, J. Franke, K. Rehman, M. Sauer, and S. Karnouskos, "I4.0-compliant integration of assets utilizing the asset administration shell," in *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2019, pp. 1243–1247.

[16] S. R. Bader and M. Maleshkova, "The semantic asset administration shell," in *International Conference on Semantic Systems*. Springer, 2019, pp. 159–174.

[17] S. Cavalieri, S. Mulé, and M. G. Salafia, "OPC UA-based asset administration shell," in *IECON 2019-45th Annual Conference of the IEEE Industrial Electronics Society*, vol. 1. IEEE, 2019, pp. 2982–2989.

[18] S. Cavalieri and M. G. Salafia, "Insights into mapping solutions based on OPC UA information model applied to the Industry 4.0 asset administration shell," *Computers*, vol. 9, no. 2, p. 28, 2020.

[19] "Examples of the asset administration shell for industrie 4.0 components - basic part," ZVEI, 2017.

[20] S. Cavalieri and M. G. Salafia, "Asset administration shell for PLC representation based on IEC 61131–3," *IEEE Access*, vol. 8, pp. 142 606–142 621, 2020.

[21] X.-L. Hoang and A. Fay, "A capability model for the adaptation of manufacturing systems," in *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2019, pp. 1053–1060.

[22] P. Marcon, C. Diedrich, F. Zezulka, T. Schröder, A. Belyaev, J. Arm, T. Benesl, Z. Bradac, and I. Vesely, "The asset administration shell of operator in the platform of Industry 4.0," in *2018 18th international conference on mechatronics-mechatronika (me)*. IEEE, 2018, pp. 1–5.

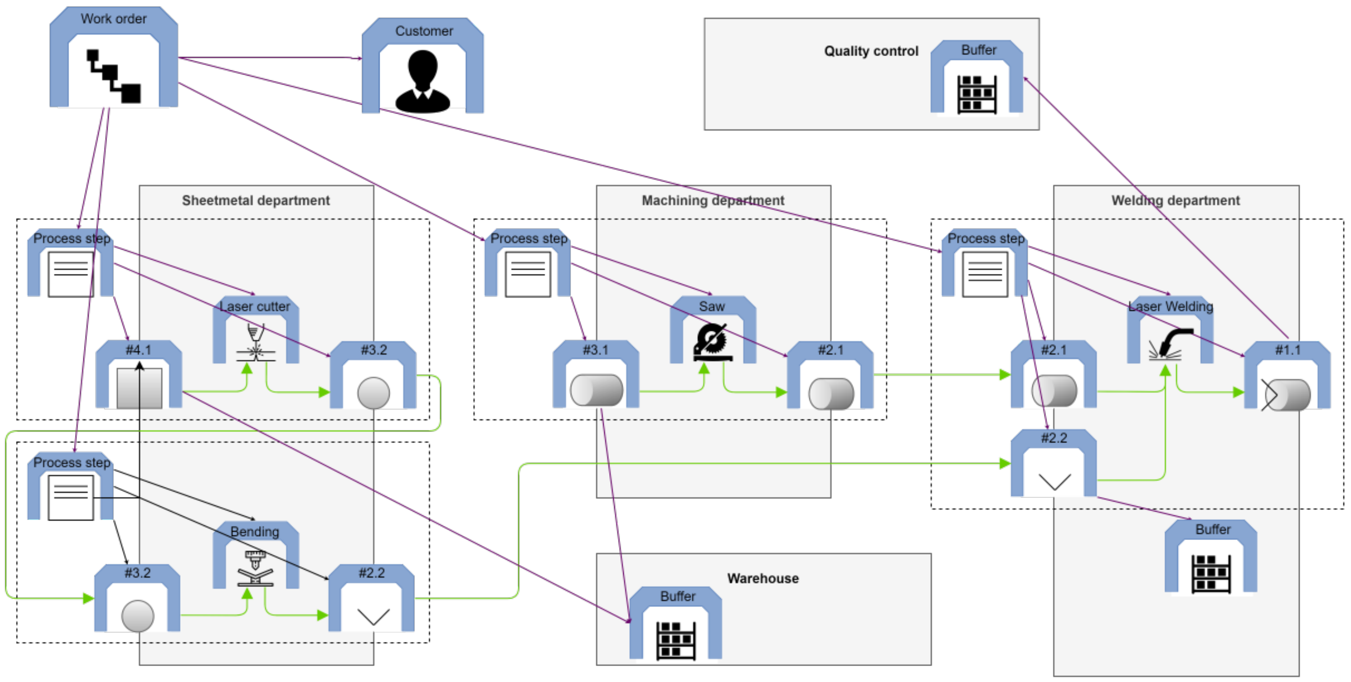


Fig. 12: Deployment of the AASs in a limited the scenario containing four process steps. The purple lines are explicitly modelled in the AAS as ReferenceElements. The green lines visualize the production flow.

[23] X. Ye, J. Jiang, C. Lee, N. Kim, M. Yu, and S. H. Hong, "Toward the plug-and-produce capability for Industry 4.0: An asset administration shell approach," *IEEE Industrial Electronics Magazine*, vol. 14, no. 4, pp. 146–157, 2020.

[24] M. Wenger, A. Zoitl, and T. Müller, "Connecting PLCs with their asset administration shell for automatic device configuration," in *2018 IEEE 16th International Conference on Industrial Informatics (INDIN)*. IEEE, 2018, pp. 74–79.

[25] "BaSys 4.0: Metamodell der Komponenten und ihres Aufbaus," BaSys 4.0, 2018.

[26] "Details of the asset administration shell - part 2," Plattform Industrie 4.0, ZVEI, 2020, version 1.0.

[27] S. R. Bader, I. Grangel-Gonzalez, P. Nanjappa, M.-E. Vidal, and M. Maleshkova, "A knowledge graph for Industry 4.0," in *European Semantic Web Conference*. Springer, 2020, pp. 465–480.

[28] C. Bezerra, F. Freitas, and F. Santana, "Evaluating ontologies with competency questions," in *2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, vol. 3. IEEE, 2013, pp. 284–285.

[29] R. d. A. Falbo, "SABiO: Systematic approach for building ontologies," in *Proceedings of the 1st Joint Workshop ONTO.COM/ODISE, co-located with the 8th International Conference on Formal Ontology in Information Systems (FOIS)*. CEUR, 2014.

[30] S. Bagui and R. Earp, *Database design using entity-relationship diagrams*. Auerbach Publications, 2012.

[31] H. Singh and S. I. Hassan, "Effect of SOLID design principles on quality of software: An empirical assessment," *International Journal of Scientific & Engineering Research*, vol. 6, no. 4, 2015.

[32] "VWS-Referenzmodellierung," Plattform Industrie 4.0, 2021.

[33] P. Drahoš, E. Kučera, O. Haffner, and I. Klimo, "Trends in industrial communication and OPC UA," in *2018 Cybernetics & Informatics (K&I)*. IEEE, 2018, pp. 1–5.

[34] N. Petersen, M. Galkin, C. Lange, S. Lohmann, and S. Auer, "Monitoring and automating factories using semantic models," in *Semantic Technology*, Y.-F. Li, W. Hu, J. S. Dong, G. Antoniou, Z. Wang, J. Sun, and Y. Liu, Eds. Cham: Springer International Publishing, 2016, pp. 315–330.

[35] J. Pfrommer, M. Schleipen, and J. Beyerer, "PPRS: Production skills and their relation to product, process, and resource," in *2013 IEEE 18th Conference on Emerging Technologies Factory Automation (ETFA)*, 2013, pp. 1–4.

[36] J. Bosak, T. McGrath, and G. K. Holman, "Universal business language v2. 0," *Organization for the Advancement of Structured Information Standards (OASIS), Standard*, 2006.

[37] L. Sakurada and P. Leitão, "Multi-agent systems to implement Industry 4.0 components," in *2020 IEEE Conference on Industrial Cyberphysical Systems (ICPS)*, vol. 1. IEEE, 2020, pp. 21–26.

[38] B. Vogel-Heuser, M. Seitz, L. A. C. Salazar, F. Gehlhoff, A. Dogan, and A. Fay, "Multi-agent systems to enable Industry 4.0," *at - Automatisierungstechnik*, vol. 68, no. 6, pp. 445–458, 2020.