# Linked Asset Administration Shells Using External Knowledge Graphs

Yuanwei Qu
Martin G. Skjæveland
Arild Waaler
*University of Oslo, Norway*
*{quy,martige,arild}@ifi.uio.no*

Andreas Schüller
*YNCORIS GmbH & Co. KG, Germany*
*andreas.schueller@yncoris.com*

Eduard Kamburjan
*IT University of Copenhagen, Denmark*
*University of Oslo, Norway*
*eduard.kamburjan@itu.dk*

***Abstract*—Systems Engineering plays a crucial role in the digitalisation of modern engineering practices, e.g., as part of Industry 4.0 and Digital Engineering. A common approach is to create a multi-faceted system model of a cyber-physical system, and then use asset models to exchange information between stakeholders about its components. However, ensuring consistency and synchronisation between these decentralised asset models without exposing the monolithic system model must be ensured. Here, we present an approach where the system model, in the form of a knowledge graph, is used to link *Asset Administration Shells* (AAS), an exchange format in Industry 4.0 based on submodel templates. This approach allows AAS to reference each other using the knowledge graph, *without exposing the graph* itself or even its vocabulary. These references are checked for consistency, generated and automatically processed by the party owning the system model. We introduce a submodel template, discuss the relation to other submodel templates, and report on its application in industrial scenarios.**

## I. Introduction

*a) Motivation.:* Engineering domains are undergoing far-reaching digitalisation that reshapes modelling practices, information exchange, and the very nature of manufactured products. To coordinate the numerous digitally connected disciplines, large-scale initiatives such as Industry 4.0, Digital Engineering, and Digital Twins, rely on *system models* based on systems engineering to describe the design and implementation of cyber-physical assets and their relations. From a Semantic Web perspective, such system models naturally embody interlinked structures,where semantic technologies can provide the formal means to express, exchange, and reason over these interconnections.

Semantic technologies have therefore gained increasing attention in system modelling. Approaches such as the Information Modelling Framework (IMF) [8], the Ontology Modelling Language (OML) [10], or those built on established formalisms such as SysML [13] exemplify this trend. While these system-level representations are valuable for engineering analysis, day-to-day collaboration between stakeholders typically occurs at a more fine-grained level: the level of individual system assets. The Asset Administration Shell (AAS) is a wide-spread serialisation and exchange format for such individual asset-level information models. AAS structures asset information into modular submodels, each typed by a submodel template, enabling stakeholders to exchange design and technical data about single components or sub-systems.

Ideally, exchanges of asset information would remain aligned with the system model. However, system models often contain sensitive design knowledge and can only be partially disclosed. As a result, asset information models may only contain a part of the information, and explicit graph structure of the system model is lost. But while the asset information models refer to single components and sub-systems, they must still refer to the other asset information models for context. For example, an asset information model for offered equipment must refer to the asset model of the requirements, or to physically adjacent equipment.

*b) Challenge.:* It is here that the challenge investigated in this work arises: *How to manage references between asset information models, if the graph which the references are based on is not disclosed?* The references between asset information models must be consistent with the system model. However, not all stakeholders have access to it. Managing a network of references bottom-up does not scale. Indeed, even the notion of consistency may not be exposed. Consider the case where a company uses an internal, company-level ontology for systems modelling. A reference of asset information model $A$ to asset information model $B$ must correspond to some path in the system model, but the labels of the path are not disclosed. A reference to the requirements must be treated differently from a reference based on physical connections.

The reference mechanisms in AAS, semantic identifiers and the Bills of Material submodel, are not sufficient to link multiple AASs through a partially disclosed system model, as we argue in section II and section IV-C. In this work, we present a general framework to link asset information models through a knowledge graph, instantiate it for AAS and demonstrate its applicability in two industrial case studies.

*c) Contributions and Structure.:* Our main contributions are (1) a general framework to link asset information models through a knowledge graph, and (2) an evaluated instantiation for the Asset Administration Shell. This article is structured as follows. We first introduce the foundations and related in systems engineering and AAS in section II, before we give

The work is partially supported by the EU Commission funded projects SM4RTENANCE (101123490) and Tec4MaaSEs (101138517).

the general theory and methodology in section III and the instantiation for AAS in section IV. Section V discusses the generation of linked AAS from an IMF system model, and section VI presents two case studies.

## II. Background and Literature Survey

*a) Systems Engineering.:* Systems engineering is a transdisciplinary and integrative approach to engineering projects [18], and model-based systems engineering is its realisation through models, e.g., architecture, simulation and knowledge models. At its core, it aims to connect different engineering disciplines by an integrative, overarching perspective over the whole design and operation lifecycle of a system.

To this end, structural models that connect and coordinate different views on a system are used. Such views can either stem from different engineering disciplines, e.g., electrical and mechanical engineering, but also from different aspects or roles. A commonly used perspective is to model function, product and location of a component separately, thus separating the requirements (in the form of the function it is designed to fulfil) from the implementation (in the form of the physical asset that is built) [11]. Another perspective may include procurement, where a function is connected with all offers and an eventual implementation.

A system model is an internal view of a system and its design. Thus, it should not be shared with parties outside the company. Yet, communication with, e.g., vendors regarding the requirements must be based on the system model. The communication itself is realised using an exchange and serialisation standard such as the Asset Administration Shell.

*b) Knowledge Graphs.:* A knowledge graph [15] is a graph representing knowledge about the domain and concrete instances. For our purposes, we rely on the W3C standards: A knowledge graph is a set of triples described by the RDF data standard and then consequently restricted using SHACL (which describes allowed and forbidden graph patterns). SPRAQL is a query language that allows, among others, to describe paths in a RDF graph and retrieve all node connected by such graphs. For example, the graph may contain concepts like `Sensor`, individuals like `sensor1`, triple that describe that their relations, such as (`sensor1,a,Sensor`), (`sensor1,nextTo,sensor2`), (`sensor2,maxTemp,100`). A SHACL shape would constrain that every sensor must have a maximal temperature (which is violated by the previous example), and a SPARQL query could retrieve all nodes with a path (`nextTo`) between them.

*c) Asset Administration Shell.:* The Asset Administration Shell (AAS) is a standardised metamodel for digitally representing assets in Industry 4.0 [2], [9], and is defined by a metamodel that introduces several key constructs [17]: An AAS represents a single asset and is composed of *Submodels*, which are modular containers for specific aspects like nameplate data, operation data or bill of materials. Dozens of predefined domain-specific aspects submodel templates are provided to support reusability. Within each submodel, *submodel elements*(e.g., properties, element collection, and element list) are used to represent concrete data or functions. The structure of an AAS is exemplified in fig. 1. Crucial for identification and semantics are:

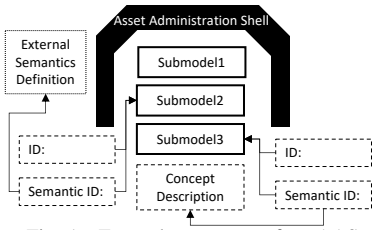

Fig. 1. Example structure of an AAS

**ID** A globally unique identifier for each submodel and submodel element.

**Semantic ID** A reference that links a submodel or element to a semantic definition. Either in an external graph or internally in a concept description.

**Concept Description** The resource in an AAS that provides the semantic meaning for a submodel or element. It can be referenced by a Semantic ID.

There are other technologies supporting manufacturing information and data exchange, such as OPC UA [19], AutomationML [1], PackML [28], and MTConnect [24]. These existing solutions are either domain-specific (e.g., AutomationML is designed to exchange structural information models during the engineering phases; however, as it is an XML-based exchange format, it is not suitable for online queries) or have limited semantics. Due to the agnostic design, the AAS ensures applicability to any asset type, in production automation, there are currently no standards in this respect that achieve the same level as the AAS [9].

### A. Literature Survey and Related Work

AAS's semantic enrichment has gained attention in recent years, driven by the need to enhance interoperability and data management in Industry 4.0 ecosystems. The Semantic Asset Administration Shell [3] introduces an RDF-based representation of the AAS data model. By mapping the XML serialisation of AAS to RDF, this work aimed to enable the use of knowledge graphs for heterogeneous data integration, validation, and reasoning. Building on this foundation, recent studies have explored applications of semantic AAS to address various aspects of data interoperability. Grangel-Gonzalez and Vidal [12] analyse a knowledge graph of Industry 4.0 standards, providing insights into how semantic technologies can standardise and integrate AAS-related information. Huang et al. [16] propose an ontology-based modelling method to enable semantic interoperability. Rongen et al. [29] propose methods for generating AAS from RDF data and using RDF data as a search index for AAS. Similarly, Louge et al. [22] suggest an event-based semantic service composition using the AAS meta-model for digital twins. This underscores the need for robust linking mechanisms to coordinate interactions between the AAS components. Several studies [31], [33] introduce

Large Language Model-based methods for AAS creation and map add semantics to the AAS models. Wawrzik et al. [32] integrate LLM-based extraction with systems engineering domain ontology to generate and quality-control engineering KGs for further usages. These efforts highlight the growing adoption of semantic technologies in AAS applications, particularly for enhancing system adaptability, but they do not fully tackle the systematic management of cross-shell references.

The challenge of managing references between submodels across different AAS shells is a significant barrier to interoperability. Gruner et al. [14] address this gap by proposing guidelines and infrastructure for cross-submodel referencing, suggesting a registry to store and resolve inter-AAS links and use Module Type Package submodel templates to connect elements across submodels. Da Silva et al. [5] develop bidirectional mappings between capability/skill models and AAS submodels, aiming to enable automated consistency checks against a knowledge graph. These contributions focus on specific model types, rather than a general framework aimed at preventing system model disclosure.

Parallel efforts have focused on using semantic-enriched AAS to support compatibility and interoperability in digital twins. Koutrakis et al. [21] explore the harmonisation of diverse AASs through intermediate model mappings, improving cross-vendor interoperability. Ocker et al. [27] use semantic AAS to conduct a compatibility check using SHACL shapes and SPARQL queries for digital twins. This work illustrates the potential of semantic technologies to ensure asset compatibility. Complementing rule-based compatibility checks, Metović et al. [23] combine SPARQL pre-filtering with KG embeddings to match heterogeneous AAS for cross-shell discovery and reuse. The work of Kamburjan et al. [20] presents a method for structural reconfiguration of digital twins using asset models in combination with semantic web technologies A survey by Beden et al. [4] provides a comprehensive overview of semantic-enriched AAS, noting that while single-shell annotation and discovery are well-supported, standardised patterns for expressing, validating, and maintaining links across multiple AASX files remain underdeveloped. Nakajima et al. [26] use OML and openCAESAR to validate SysML models through DL reasoning, but their approach targets a single system model rather than linking distributed asset models. Related work also applies ontology-based reasoning to check SysML model consistency, but it does not address selective disclosure or cross-AAS references [25].

Current connections between AAS and semantic technologies focus on RDF representations, mapping AAS to RDF schemas, or generating AAS models from RDF data. The AAS design, using semantic identifiers and hierarchical Submodels (e.g., BoM) only achieves vocabulary-level interoperability and fails to capture semantic relationships. Complementing AAS-centric semantic representations, the Function–Behaviour–Structure (FBS) framework [11] shows how a structural module mediates between a system's functions and behaviours. This idea motivates the use of structural modules to define semantically typed links across asset models. Our proposed architecture builds on these efforts, overcoming their limitations with three innovations: (1) decoupling reference semantics from AAS serialisation formats for flexibility, (2) adding consistency checks, and (3) enabling queries across AAS instances by using graphs as index structures. This blends AAS's industrial strengths with Semantic Web techniques for a scalable and standardised solution that aligns with the intention to achieve interoperability in industrial ecosystems.

*a) Running Example:* For the remainder of this work, we use a simplified version of the industrial case study introduced in section VI-A. Consider a company A that designs large-scale manufacturing equipment. For one such design, company A has a system model for all the components and their requirements in a graph structure (fig. 2). When one of these components must be procured externally, the relevant requirements are extracted from the system model into an AAS `R` and sent to vendor B. The vendor responds with a proposal AAS `P`, that must correctly refer to `R` in the system model, even though the vendor has no access to that model.

## III. Linked Asset Models

Before turning to Knowledge Graph and AAS specifics, we introduce an abstract, formal framework to link asset information models over an arbitrary graph-based system model. This generic formulation can be instantiated for different graph models and different asset information modelling approaches. It also clarifies the general requirements that any concrete instantiation must satisfy.

*a) Components.:* We use an arbitrary graph model as the system model. It is not important whether this graph is formed directly by graph data, a system model in the sense of, e.g., SysML, IMF, and other component languages, or graph data generated out of it. As basic notation, given a set $X$ we denote with $X^*$ sequences of arbitrary length with elements from $X$ and with $\mathsf{Tree}(X)$ the set of finite trees over $X$, and given a tree $T$ with $Seq(X)$ the set of all paths in $T$.

*Definition 1 (System Model Graph):* Let $V$ be the set of all vertices, $L$ a label set for the edges, and $\mathbf{D}$ a set of data values, for example, integers and other literals. A *system model graph* is a tuple $\mathcal{S} = (V_\mathcal{S}, E_\mathcal{S})$ with $V_\mathcal{S} \subseteq V$, $E_\mathcal{S} \subseteq V_\mathcal{S} \times L \times V \cup V_\mathcal{S} \times L \times \mathbf{D}$.

All information needed for system modelling is in the label and eventual restrictions on the structure of the graph. If there is a path $P \in E_\mathcal{S}^*$ from $v \in V_\mathcal{S}$ to $w \in V_\mathcal{S}$ in $\mathcal{S}$, then we write $v \xrightarrow[\mathcal{S}]{P} w$. If $\mathcal{S}$ is understood, then we write $v \xrightarrow{P} w$.

*Example 1:* Consider a system model with one root node for the process design, one system design that composes two components and one system implementation that only realises one of the system design components. Figure 2 shows the system graph visually, with `/graph/n/`$x$ being the names of the nodes. The design components model the requirements (left part), in the example maximal and minimal temperature during operations, while the implementation components express the data sheet of a concrete component (right part).

An *asset information model* is a model describing information and data of the asset. It is independent of the system model

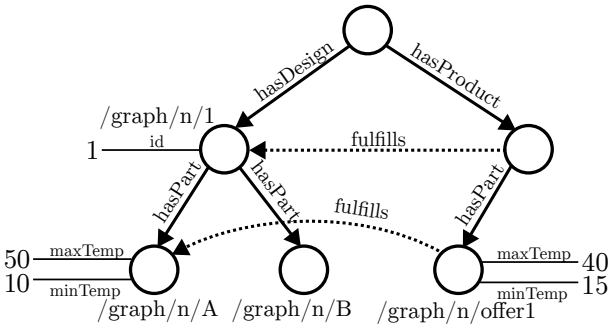


Fig. 2. System graph for our running example.

graph, but we introduce *anchors*: An anchor is a reference from an asset information model into a system model graph, and refers to what part of the system the asset in question represents. Note that the anchor refers to a name or identifier, but a node. How a node in a system models refers to the identity of the asset is determined by the specification.

*Definition 2 (Asset Information Model):* Let $Id$ be a set of identifiers, $D$ a set of data labels, and $R$ a set of reference labels.

An *asset information model* (AIM) is a tuple $\mathcal{A} = (id_{\mathcal{A}}, rng_{\mathcal{A}}, achr_{\mathcal{A}}, refs_{\mathcal{A}})$ where $id \in Id$, $achr \in V$, $rng \subseteq D \times \mathbf{D}$, and $refs \subseteq R \times Id$.

An asset information model has an identifier ($id_{\mathcal{A}}$) and the aforementioned anchors ($achr_{\mathcal{A}}$). Additionally, it has all the data about the information model as a map from data labels to data values. Finally, it can refer to other AIMs.

*Example 2:* The AIM that describes the system components design of nodes `/graph/n/A` and `/graph/n/offer1` are as follows. Note that the labels `maxTempD`, `minTempD`, and `for` are not from the vocabulary of the system graph (fig. 2).

$$\mathcal{A}_A = \big(A, \{(\texttt{maxTempD}, 50), (\texttt{minTempD}, 10)\}, \texttt{/graph/n/A}, \emptyset\big)$$
$$\mathcal{A}_{offer} = \big(O, \{(\texttt{maxTempD}, 40), (\texttt{minTempD}, 15)\}, \texttt{/graph/n/offer1}, \{(\texttt{for}, A)\}\big)$$

We can now formulate the main challenge in the introduced model: *Given a system model graph and a set of AIMs, how to relate the data and references of the asset information models with the system model graph? In particular, how to establish a notion of consistency based on this relation?*

The relation of the system model graph and the asset information models is highly application specific, but we identify two general needs: (1) References between asset information models must correspond to the right kind of path between their anchors, and (2) the same must hold for data and paths between an anchor and data values. For these purposes, we introduce two *libraries*, which map reference labels to path labels and data labels to tree labels.

*Definition 3:* A reference library is a mapping from reference labels to sequences of edge labels. $\mathcal{L} : R \to L^*$.

Intuitively, it expresses that if there is a reference between two asset information models labelled with $r \in R$, then there must be a path labelled $\mathcal{L}(r)$ between their anchor nodes. In contrast, the data reference library maps to tree, and intuitively expresses that the value with label $d$ can be computed from the leaf data reachable from the anchor node via $\mathcal{D}(d)$, by applying function $\mathcal{F}(d)$.

*Definition 4 (Data Reference Library):* Let $arity : \mathsf{Tree}(L) \to \mathbb{N}$ be the number of leaves in a tree. A data reference library $\mathcal{D} : D \to \mathsf{Tree}(L)$ is a mapping from data labels to trees of edge labels. The computation function $\mathcal{F} : D \to \mathbf{D}^{\mathbb{N}} \to \mathbf{D}$ maps each data label to a function $\mathcal{F}(d) : \mathbf{D}^{\mathcal{D}}(d) \to \mathbf{D}$.

We refer to a triple $(\mathcal{L}, \mathcal{D}, \mathcal{F})$ simply as *information library*.

*Example 3:* Continuing our running example, we have the following information library. In our example, it is just a label renaming, and the data is extracted directly, hence the identities in the computation function.

$$\mathcal{L} = \{\texttt{for} \mapsto \texttt{fulfills}\}$$
$$\mathcal{D} = \{\texttt{maxTempD} \mapsto \texttt{maxTemp}, \texttt{minTempD} \mapsto \texttt{minTemp}\}$$
$$\mathcal{F}(\texttt{maxTempD})(x) = x \qquad \mathcal{F}(\texttt{minTempD})(x) = x$$

*b) Operations.:* Two operations are required in our framework: (1) Given a system model graph and an information library, generate a set of AIMs. (2) Given a system model graph, an information library and a set of asset information models, determine whether they are consistent.

We consider consistency and formalise the intuition we gave above. We allow the use of different system model graphs. This mirrors that an asset information model may model only one asset, which is, however, used in several system models, e.g., of different companies. Let $I, J$ be finite sets.

*Definition 5 (Consistency):* Let $(\mathcal{S})_{j \in J}$ be a set of system model graph with pairwise disjoint vertex sets, $(\mathcal{L}, \mathcal{D}, \mathcal{F})$ an information library, and $(\mathcal{A})_{i \in I}$ as set of asset information models. Let $\mathcal{S}(id)$ be the system model graph such that $achr \in \mathcal{S}(id)$ for the $achr$ of the asset information model with identifier $id$.

We say that a set of system model graphs and a set of asset information models are *consistent*, denoted $(\mathcal{S}_j)_{j \in J} \models_{(\mathcal{L},\mathcal{D},\mathcal{F})} (\mathcal{A}_i)_{i \in I}$, if the following conditions hold for every $i \in I$.

1) Each anchor node $achr_{\mathcal{A}}$ indeed is a valid node in a system model graph: $\exists j.\ achr_{\mathcal{A}} \in V_{\mathcal{S}_j}$.
2) The identifier and anchor node of each asset information model are unique.
3) If there is a reference label $r$ between two asset information models that are both anchored in the same system model graph, then there is a path labelled $\mathcal{L}(r)$ in this graph according to the library.
4) If there is a data level $d$ with data value $\mathbf{d}$ in an asset information model, then this value is computed by applying $\mathcal{F}$ to $\mathcal{D}(d)$.

Consistency is also a constraint on the instantiation of our framework: We require that generating a set of asset information models using the libraries results in a consistent set.

*Definition 6:* Let $\mathcal{S} = (V_{\mathcal{S}}, E_{\mathcal{S}})$ be a system model graph, $(\mathcal{L}, \mathcal{D}, \mathcal{F})$ be an information library and $T \subseteq V_{\mathcal{S}}$ be a set

of nodes. The generated set of asset information models is defined as follows, where each $id_{\mathcal{A}}$ is a fresh identifier, and $refs_{\mathcal{A}}$ and $rng_{\mathcal{A}}$ are directly induced by the conditions above. By construction, generated AASs are consistent with the single system model graph that is their origin: $\{\mathcal{S}\} \models_{(\mathcal{L},\mathcal{D},\mathcal{F})} [\![targets]\!]_{\mathcal{S},(\mathcal{L},\mathcal{D},\mathcal{F})}$.

## IV. Linked Asset Administration Shells

We now instantiate our concepts of linking AIMs through a system model graph for AAS. First, we give the structure of the reference submodel template, before we clarify the relation to other submodels. The information library is outside the submodel; we discuss its implementation afterwards.

### A. Reference Library

The reference library (cf. definition 3) maps reference labels between asset information models to path labels between the anchor nodes. For AAS, a reference label is a concept description, and the reference library is an endpoint that (a) lists the reference labels allowed for a certain linked AAS application and (b) maps the reference label to a SPARQL query that allows checking whether two nodes are connected by a path described by this query.

The query is not exposed to the outside. This is to realise an additional layer of information hiding: Queries expose the vocabulary and information about the structure of the knowledge graph. The vocabulary can, for example, expose terminology of the application ontology used internally by the organisation.

We assume that a reference library is identified by a URI `ref.lib`. Given a reference label $r$, the corresponding concept description has the ID `ref.lib/`$r$ and querying this ID returns an AAS containing only this concept description. The only restriction on this concept description is that it has exactly one IEC 61360 Data Specification that specifies its data type as a String/URI.

### B. Reference Submodel Template

A submodel template is submodel instance without concrete values. It is split into mandatory and optional parts, which we describe for our *reference submodel*.

*a) Mandatory Elements.:* The following elements are mandatory and model the following information.

**id** The submodel, like any submodel, must have its own identifier. This maps to $id_{\mathcal{A}}$ in our abstract definition.

**graph_id** This is the URI of the system model graph. Exactly one property of this type must be provided. This is implicit in our abstract definition as part of $achr_{\mathcal{A}}$.

**node_id** This is the URI of the anchor node for the system model graph. Exactly one property of this type must be provided. This maps to $achr_{\mathcal{A}}$ in our abstract definition.

**managed_element** This is the URI of either a submodel or an AAS, whose relations are managed by this submodel. Exactly one property of this type must be provided. This maps to the identity, but not the content of $rng_{\mathcal{A}}$. In practice, we expect that the managed_element will either be the technical data submodel (see below) of an AAS or the AAS that contains this submodel. The managed_element realizes $rng_{\mathcal{A}}$.

*b) Optional Elements.:* The remaining elements are realising $refs_{\mathcal{A}}$. For each reference, a property is added to the submodel, which has the semantic id of the concept description. The concept description `fulfillsCD` has the id `ref.lib/fulfils` and describes that the value must be an URI. The property itself has the semantic id for `fulfillsCD` and as its value the id of the other asset.

### C. Discussion

*a) Consistency.:* To check the consistency of a set of AAS w.r.t. a system graph $G$, we must relate them to the system model graph. As discussed above, each reference $r$ has the form `ref.lib/`$r$ and identifies a SPARQL query in two variables *?from* and *?to*. Given an AAS $\mathfrak{A}$ that has a reference submodel, we can extract a partial abstract asset information model $\mathcal{A}_{\mathfrak{A}}$ as follows.

- $id_{\mathcal{A}_{\mathfrak{A}}}$ is the id of the reference submodel.
- $achr_{\mathcal{A}_{\mathfrak{A}}}$ is the node_id of the reference submodel.
- $refs_{\mathcal{A}_{\mathfrak{A}}}$ is defined as $(r, id)_I$, the set of optional properties in the reference submodel that have a semantic id of the form `ref.lib/`$r$ and the value $id$.
- $rng_{\mathcal{A}_{\mathfrak{A}}}$ is the id of the managed_submodel.

Given a set $(\mathfrak{A})_I$ of AAS with reference submodels all pointing to the same graph $G$, we say that it is *reference-consistent* if the following conditions hold: (a) All $rng_{\mathcal{A}_{\mathfrak{A}}}$ ids are different. (b) All $achr_{\mathcal{A}_{\mathfrak{A}}}$ URIs are different. (c) For each reference $(r, id) \in refs_{\mathcal{A}_{\mathfrak{A}}}$ in an abstract AIM with id $uri'$, there is a reference submodel with managed_id $id$ and id $uri$, and the query $\mathcal{L}(r)[?\texttt{from} \mapsto uri', ?\texttt{to} \mapsto uri]$ returns true. This can be realised by an endpoint without exposing the query.

With regard to the conditions (1)–(4) for general consistency in definition 5, conditions (a) and (b) map to condition (2), while condition (c) maps to condition (3). Condition (1) holds trivially by construction of the set, and condition (4) is related to data – we describe below how to extend reference-consistency to full consistency.

*b) Relation to other Submodels.:* The reference submodel is used together with other submodels and is partially redundant in its content and scope with the *Hierarchical Structures enabling Bills of Material*[1] (BoM). However, as AAS is an exchange format and not a modelling formalism – as AAS are exported from ground truth models, redundancy in the serialisation does not incur technical depth in further management.

BoM aims to describe hierarchical structures in assets and is based on the Entities and Relationships part of the AAS metamodel, motivated by the restriction of the AAS metamodel that submodels and AAS cannot be nested. An example is given as part of fig. 4, where we describe how

[1] https://industrialdigitaltwin.org/wp-content/uploads/2023/04/IDTA-02011-1-0_Submodel_HierarchicalStructuresEnablingBoM.pdf

to relate BoM and the Reference Submodel. The obvious difference to the Reference Submodel is that it only provides a single partOf relation and its inverse. A less direct difference is that it is fully exposing the structure – it is not possible to achieve the level of flexibility and information handling that an external knowledge graph provides.

*c) Running Example.:* Figure 2 shows the system graph for the situation of our running example. The left part of the system graph describes the requirements for a piece of equipment, while the right describes the draft for the equipment. This results in two AAS, shown in fig. 3. The left AAS is anchored in the requirements and refers to the right AAS as its potential implementation. The right AAS, anchored in the implementation, refers to the left AAS as the requirement it fulfils. Both references map to different paths through different parts of the graph. In particular, the potential implementation contains nodes that should not be exposed, as they model information about currently ongoing purchase processes and internal terminology.

*d) Bills-of-Material.:* Consider again the same system graph as before. Figure 4 shows how this hierarchical structure is expressed using the reference submodel to manage references. In contrast, the BoM submodel would contain information about three assets and the hierarchy directly as in the right of fig. 3. In this case, the AAS for the node `/graph/n/A` is different – the information is always in the system model, and the AASs are generated for each use case specifically.

The hierarchy is explicit and does not refer to the system graph. Why the latter can be added using semantic ids, the former is fixed: the BoM submodel does not describe other labels for relations, and introducing them to AAS would require duplicating the complete vocabulary of system modelling.

## V. Generation

In the following, we describe how to instantiate the framework using the Information Modelling Framework (IMF) [8] as the framework for the system model graph. IMF is a language to describe systems in terms of blocks, ports and connections, where each such element has an aspect such as product or function. IMF has a semantics in terms of a knowledge graph, which serves as the system model graph. The goal is to extract from the system model all information required for a complete data exchange and for reference-consistency checking.

We focus on the instantiation of the computation function and data reference library for a single AAS, as the references are handled generically. It must expose as little as possible of the structure of the system model graph, and be configurable for different situations, as the degree of exposure depends on the recipient of the exported AAS.

*a) Overview.:* The translator takes as input an RDF graph that conforms to the IMF metamodel as described by the IMF OWL ontology and SHACL shapes. The input graph represents blocks and terminals which are associated with aspects, attributes, relationships and classifiers. Each `imf:Block` becomes its own AAS instance. The block's IRI is recorded as the global asset identifier of the resulting AAS, i.e., its anchor id as described in section III. For each block, the translator generates a Technical Data submodel containing all extracted data from the IMF model. Additional auxiliary semantic IDs are used to record further information from the IMF model, such as the modelled aspect.

*b) Technical Data Submodel.:* The technical submodel for a given block contains information about the attached attributes, terminals, and dependencies. The block and each such attached component have their own structure inside the same AAS. Note that they are not connected using the reference submodel, as they are directly related in the export.

Each IMF attribute associated with a block or terminal is translated into an AAS property, i.e., a data label in our framework. Literal values and datatypes are preserved and translated into the correct AAS datatype format. Attribute metadata, such as units and attribute qualifiers, are preserved through the use of AAS qualifiers. The IRI of the attribute itself becomes the AAS element's semantic ID, thereby preserving the semantic grounding of the attribute in the IMF model. This is analogous to the anchor id for blocks.

Terminals, i.e., elements of `imf:hasTerminal`, are treated analogously: each terminal becomes a collection containing its attributes. This yields a faithful structural representation of the system interfaces of a block.

*c) Dependencies.:* Dependency relations between blocks, such as requirement–solution relations (`imf:fulfills`), part–whole breakdowns (`imf:partOf`), and specialisation (`imf:specializationOf`), are represented through a recursive structure. This produces a tree-shaped local view of the relevant neighbourhood around each block in the IMF graph. For each related entity (1) a SubmodelElementCollection is created containing the related entity's attributes, classifiers, type and aspect annotations, and a semantic ID referring to the entity's IRI, and (2) the collection contains a nested Dependencies collection, to which the same transformation is applied recursively. However, because the dependency design mirrors the underlying graph through a recursive expansion, the resulting tree inevitably discloses all connected relations and neighbouring entities. This exposure is even more direct than BoM as it reveals the full local neighbourhood of the target entity. This reinforces the need for our approach, which provides controlled, semantically grounded references without exposing the underlying system model.

## VI. Application and Adoption

The described method was applied and evaluated in ongoing industrial cases involving the exchange of information models using AAS.

### A. Selective Disclosure of System Context

The manufacturing industry increasingly recognises the importance of governed and interoperable asset-information exchange. In the Tec4MaaSEs project, which develops a digital-twin platform and governance services for manufacturing, we apply the use of AAS for managing asset information

AAS Design

| Tech. SM Id | /aas/dat/A |
|---|---|
| Ref. SM Id | /aas/ref/A |
| graph_id | /graph/ |
| node_id | /graph/n/A |
| managed. | /aas/dat/A |

AAS Offer

| Tech. SM Id | /aas/dat/B |
|---|---|
| Ref. SM Id | /aas/ref/B |
| graph_id | /graph/ |
| node_id | /graph/n/offer1 |
| managed. | /aas/dat/B |
| fulfills-rel. | /aas/dat/A |

- EntryNode: `/aas/1`
  - hasPart: `/aas/A`
  - hasPart: `/aas/B`
- Node: `/aas/A`
  - partOf: `/aas/1`
- Node: `/aas/B`
  - partOf: `/aas/1`

Fig. 3. Using the reference submodel to link AAS Design (left) and AAS Offer (middle). On the right we see a BoM submodel for the overall structure.

AASA

| Tech. SM Id | /aas/dat/A |
|---|---|
| Ref. SM Id | /aas/ref/A |
| graph_id | /graph/ |
| node_id | /graph/n/A |
| managed. | /aas/dat/A |
| partOf-rel. | /aas/dat/1 |

AASB

| Tech. SM Id | /aas/dat/B |
|---|---|
| Ref. SM Id | /aas/ref/B |
| graph_id | /graph/ |
| node_id | /graph/n/B |
| managed. | /aas/dat/B |
| partOf-rel. | /aas/dat/1 |

AAS1

| Tech. SM Id | /aas/dat/1 |
|---|---|
| Ref. SM Id | /aas/ref/1 |
| graph_id | /graph/ |
| node_id | /graph/n/1 |
| managed. | /aas/dat/1 |
| part-rel. | /aas/dat/A |
| part-rel. | /aas/dat/B |

Fig. 4. Three AASs encoding a hierarchical structure.

in the Aibel case. Aibel is an engineering, procurement, and construction contractor currently studying or developing a seawater cooling system for a hydrogen production facility. A key component of this system is a flow meter, which ensures accurate monitoring of seawater flow and is essential for safe and efficient operation. Because Aibel does not manufacture flow meters, it must procure one from external suppliers.

In this process, Aibel provides the supplier with an AAS-based specification of the required flow meter, and the supplier responds with an AAS describing a device they believe meets the requirements.A central challenge is that Aibel must give the supplier enough system context to select an appropriate flow meter, yet avoid revealing sensitive internal design details. The standard BoM submodel, would reveal the entire part-of hierarchy—for example, that the flow meter belongs to a pump system, which together with two others forms the seawater lift pump system, which in turn is part of the seawater cooling system. Aibel instead aims to share only the top-level seawater cooling system information together with the detailed flow-meter specification.

This challenge is addressed using a reference submodel: Aibel shares only the required flow-meter specifications together with the top-level specifications of the seawater cooling system. The part-of relation is treated as a transitive relation defined in a concept description, whose semantic ID points to an externally defined ontology. Figure 5 shows an example offer AAS about a magnetic flow-meter from a supplier, where the technical data provided fulfils the requirements specified in Aibel's technical-data submodel. Although we cannot publish the detailed technical data, the example reflects an actual industrial case and has been agreed by domain experts.

### B. Data Spaces

A data space is a *"distributed system defined by a governance framework that enables secure and trustworthy data transactions between participants while supporting trust and data sovereignty"* [6], a concept that is currently being implemented in several European initiatives for different industries within the European Union, especially for engineering [30]. It allows the exchange of data in *data products*, *"standardised data units packaging relevant data resources and services into a consumable form complying with data product specifications"* [7].

In the SM4RTENANCE project, which develops the European data space for smart manufacturing, we have explored the use of AAS as the basis for data products in the Itema use case. SM4RTENANCE builds a framework for predictive-maintenance data sharing, aligning industrial stakeholders around common models, vocabularies, and interoperability practices across the asset lifecycle. Itema produces weaving machines and collects data for, among others, predictive maintenance. Both the weaving machines, including sensors and endpoints to retrieve their data, and the digital cloud infrastructure for data structure, are described in an IMF system model and available as an RDF system model graph.

AASs are used as the basis for the data products: the endpoints that make the data available to the data space. These data products also require selective disclosure. While AASs may refer to each other the system model of the machines and the cloud infrastructure must not be imposed.

## VII. Conclusion

In the highly digitalised and data-driven engineering practices of today, it is important to exchange information and data about assets, while controlling the disclosure of the underlying system model and adopting standardised exchange and modelling frameworks. We presented a general method for linking asset information models through a system model in the form of a knowledge graph and demonstrated its AAS instantiation. The approach enables asset models to reference each other consistently without exposing the underlying graph

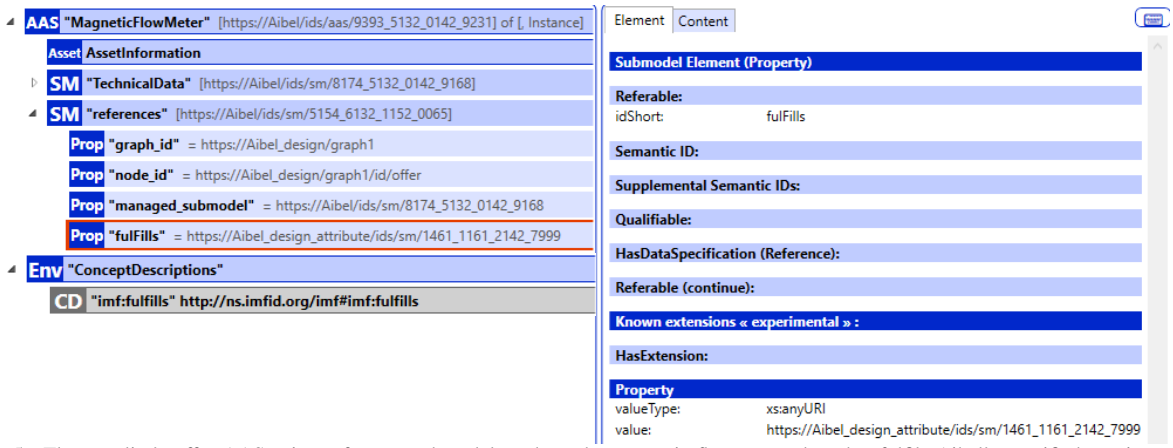


Fig. 5. The supplier's offer AAS using reference submodel to show the magnetic flow-meter data that fulfils Aibel's specified requirements.

or vocabulary, addressing limitations of existing AAS mechanisms for partially disclosed system models. Future work includes richer constraint libraries and broader evaluations across additional domains.

## References


[1] AutomationML: What is automationml? Tech. rep., AutomationML association (2025), https://www.automationml.org/about-automationml/automationml/

[2] Bader, S., Barnstedt, E., Bedenbender, H., Berres, B., Billmann, M., Ristin, M.: Details of the asset administration shell-part 1: the exchange of information between partners in the value chain of industrie 4.0 (version 3.0 rc02) (2022)

[3] Bader, S.R., Maleshkova, M.: The semantic asset administration shell. In: SEMANTiCS. pp. 159–174. Springer (2019)

[4] Beden, S., Cao, Q., Beckmann, A.: Semantic asset administration shells in industry 4.0: A survey. In: ICPS. IEEE (2021)

[5] Da Silva, L.M.V., Köcher, A., Gill, M.S., Weiss, M., Fay, A.: Toward a mapping of capability and skill models using asset administration shells and ontologies. In: ETFA. pp. 1–4. IEEE (2023)

[6] Data Space Support Centre: Glossary: Core concepts (2023), https://dssc.eu/space/Glossary/176554052/2.+Core+Concepts

[7] Data Space Support Centre: Glossary: Data products and services (2023), https://dssc.eu/space/Glossary/176554115/6.+Data+products+and+services

[8] DNV: Dnv-rp-0670 asset information modelling framework. Tech. rep., Det Norske Veritas (2024), https://www.dnv.com/digital-trust/recommended-practices/asset-information-modelling-dnv-rp-0670/

[9] EclipseBaSyx: About asset administration shells. Tech. rep., Eclipse BaSyx, Fraunhofer Institute for Experimental Software Engineering IESE (2025), https://wiki.basyx.org/en/latest/content/user_documentation/concepts%20and%20architecture/aas_overview.html

[10] Elaasar, M., Rouquette, N.: Ontological modeling language v2. Tech. rep., OpenCaesar (2025), https://www.opencaesar.io/oml/

[11] Gero, J.S., Kannengiesser, U.: The situated function–behaviour–structure framework. Design Studies **25**(4), 373–391 (2004). https://doi.org/https://doi.org/10.1016/j.destud.2003.10.010

[12] Grangel-González, I., Vidal, M.E.: Analyzing a knowledge graph of industry 4.0 standards. In: Web Conference. pp. 16–25 (2021)

[13] Group, O.M.: Systems modeling language (sysml) version 1.6. Tech. rep., Object Management Group (2024), https://sysml.org/

[14] Grüner, S., Hoernicke, M., Stark, K., Schoch, N., Eskandani, N., Pretlove, J.: Towards asset administration shell-based continuous engineering in process industries. at-Automatisierungstechnik **71**(8), 689–708 (2023)

[15] Hogan, A., et al.: Knowledge graphs. ACM Comput. Surv. **54**(4), 71:1–71:37 (2022). https://doi.org/10.1145/3447772

[16] Huang, Y., Dhouib, S., Medinacelli, L.P., Malenfant, J.: Enabling semantic interoperability of asset administration shells through an ontology-based modeling method. In: MODELS-C. pp. 497–502 (2022)

[17] IDTA: Specification of the asset administration shell part ersion 1.0rc03 1: Metamodel. Tech. rep., Industrial Digital Twin Association: IDTA (2023), https://industrialdigitaltwin.org/wp-content/uploads/2023/06/IDTA-01001-3-0_SpecificationAssetAdministrationShell_Part1_Metamodel.pdf

[18] INCOSE: Systems engineering 2035 vision (2022), https://incose.org/publications/se-vision-2035

[19] Inray: What is opc ua? a practical introduction. Tech. rep., Inray Industriesoftware GmbH (2025), https://www.opc-router.com/what-is-opc-ua/

[20] Kamburjan, E., Klungre, V.N., Schlatte, R., Tarifa, S.L.T., Cameron, D., Johnsen, E.B.: Digital twin reconfiguration using asset models. In: ISoLA. pp. 71–88. Springer (2022)

[21] Koutrakis, N.S., Gowtham, V., von Pilchau, W.B.P., Jung, T.J., Polte, J., Hähner, J., Corici, M.I., Magedanz, T., Uhlmann, E.: Harmonization of heterogeneous asset administration shells. Procedia CIRP **107** (2022)

[22] Louge, T., Araghi, S.N., Karray, M.H., Sarkar, A.: Events-based semantic services composition in industry 4.0 using asset administration shell meta-model for digital twins. Expert Systems with Applications p. 126641 (2025)

[23] Metovic, A., Maisch, N., Ajdinović, S., Lechler, A., Wortmann, A., Riedel, O.: Industrial semantics-aware digital twins: A hybrid graph matching approach for asset administration shells. Tech. rep., University of Stuttgart (2025), https://awortmann.github.io/preprints/

[24] MTConnect: About asset administration shells. Tech. rep., MTConnect Institute (2025), https://www.mtconnect.org/

[25] Nakajima, Y., Katsumata, H., Komatsu, Y., Elaasar, W., Wagner, D.: opencaesar application to power balance analysis in early space mission formulation. In: onto:Nexus. IEEE (2025)

[26] Nakajima, Y., Wada, A., Komatsu, Y., Elaasar, W., Wagner, D.: Power of a reasoner: Model validation for sysml model using opencaesar. In: onto:Nexus. IEEE (2025)

[27] Ocker, F., Vogel-Heuser, B., Schön, H., Mieth, R.: Leveraging digital twins for compatibility checks in production systems engineering. In: IEEM. pp. 103–107. IEEE (2021)

[28] OMAC: What is PackML? Tech. rep., The Organization for Machine Automation and Control (2025), https://www.omac.org/packml

[29] Rongen, S., Nikolova, N., van der Pas, M.: Modelling with aas and rdf in industry 4.0. Computers in Industry **148**, 103910 (2023)

[30] Schueller, A.: Process x – a data space for the process industry. WPT – Fachportal für Prozesstechnik (2025). https://doi.org/10.1002/cite.202500007

[31] Shi, D., Meyer, O., Oberle, M., Bauernhansl, T.: Dual data mapping with fine-tuned large language models and asset administration shells toward interoperable knowledge representation. Robotics and Computer-Integrated Manufacturing **91**, 102837 (2025)

[32] Wawrzik, F., Rafique, K.A., Urimubenshi, T., Grimm, C.: Kgg4se: A knowledge graph generation framework for systems engineering. Semantic Web Journal (2024)

[33] Xia, Y., Xiao, Z., Jazdi, N., Weyrich, M.: Generation of asset administration shell with large language model agents: Towards semantic interoperability in digital twins in the context of industry 4.0. IEEE Access (2024)