MODAPTO [101091996]: Modular Manufacturing and Distributed Control via Interoperable Digital Twins

13.2.2 AAS metamodel components and data structures

2025-08-11

Introduction

The AAS metamodel components and data structures define the formal specification for organizing, storing, and exchanging asset information within Industry 4.0 systems. These components provide the building blocks for creating consistent, interoperable digital representations that support MODAPTO design reconfiguration capabilities. Understanding the metamodel structure is essential for implementing effective asset digitalization and ensuring semantic consistency across diverse manufacturing environments.

The Identifiable class serves as the base class for all AAS components that require unique identification within the global namespace. This class provides the fundamental identification attribute that uses International Resource Identifiers (IRIs) to ensure globally unique addressing of AAS components. The identification system supports both absolute IRIs that specify complete resource addresses and relative identifiers that can be resolved within specific contexts. This flexible identification approach enables distributed AAS implementations while maintaining precise component referencing.

The Referable class extends Identifiable to provide human-readable naming and description capabilities that support user interaction and documentation requirements. Referable components include short names for programmatic access, display names for user interfaces, and descriptions that explain component purpose and usage. Multi-language support enables international deployment by allowing names and descriptions in multiple languages while maintaining consistent semantic meaning across different linguistic contexts.

The HasSemantics interface provides the mechanism for linking AAS components to external semantic definitions through concept descriptions. This linkage ensures that data has well-defined meaning that can be consistently interpreted by different systems and applications. Each submodel is described by several properties defined by a unique global identifier and a set of well-defined attributes, with AAS attributes including preferred name, symbol, unit of measure, and definition, and existing IEC and ISO standards considered for use within the AAS.

Semantic references support both standard vocabularies and custom ontologies to accommodate diverse industrial domains.

The AssetAdministrationShell class represents the top-level container that encapsulates all information about a specific asset. This class includes administrative information such as revision history, responsible parties, and lifecycle status. The shell provides references to associated assets and contains collections of submodels that organize asset information into logical groupings. Version management capabilities support evolution of asset descriptions while maintaining historical information and enabling rollback when necessary.

The Asset class represents the physical or logical entity being described by the AAS, including manufacturing equipment, software systems, documents, or business processes. Assets are categorized as either Type assets that represent design templates or Instance assets that represent specific implementations. From the manufacturer’s point of view the

asset is a product, with the manufacturer managing different asset types that have a history with different versions.

The asset class includes identifying information such as manufacturer data, serial numbers, and product classifications that enable precise asset identification and management.

The Submodel class provides the primary mechanism for organizing asset information into domain-specific collections. Submodels contain SubmodelElements that represent individual data points, functions, or relationships related to the asset. Common submodel types include digital nameplate information, technical specifications, operational parameters, maintenance data, and business relationships. The modular submodel structure enables flexible composition of asset descriptions while maintaining semantic consistency and supporting reuse across similar assets.

SubmodelElement classes define the various types of information that can be contained within submodels. Property elements represent simple data values with associated metadata such as data type, units of measure, and semantic references. MultiLanguageProperty elements support internationalization by providing property values in multiple languages. Range elements specify minimum and maximum values for numeric properties. File elements reference external documents or media files that provide additional asset information.

Collection elements group related SubmodelElements into logical sets that can be processed as units. OperationElements define functions that can be invoked to control asset behavior or request information. These elements include input parameters, output parameters, and in-out parameters that support bidirectional data exchange. The operation model enables AAS to represent not just static asset information but also dynamic behaviors and control capabilities.

ReferenceElement classes provide mechanisms for creating relationships between different AAS components or external resources. These references can point to other SubmodelElements, complete Submodels, or external assets and systems. The reference system supports both internal navigation within AAS structures and external integration with enterprise systems and partner organizations.

DataSpecification classes define templates for common types of asset information that promote consistency and interoperability across different implementations. These specifications include predefined property sets for common industrial domains such as mechanical engineering, electrical systems, and process control. Data specifications can be extended or customized to support specific industry requirements while maintaining compatibility with standard tooling and applications.

Qualifier classes provide additional metadata and constraints that refine the interpretation and usage of AAS components. Qualifiers can specify data quality indicators, access permissions, lifecycle states, and other contextual information that affects how components should be processed or displayed. The qualifier mechanism enables fine-grained control over component behavior while maintaining metamodel simplicity.

ConceptDescription classes provide semantic definitions for properties, operations, and other elements used within AAS implementations. These descriptions link to external vocabularies, standards, or ontologies that define precise meaning and usage guidelines. The concept description system enables semantic interoperability by ensuring that different AAS implementations can consistently interpret shared concepts and data structures.

The constraint system within the metamodel defines rules and validations that ensure AAS instances comply with structural and semantic requirements. Constraints can specify required attributes, valid value ranges, relationship cardinalities, and other rules that govern valid AAS structures. Automated validation tools use these constraints to verify AAS compliance and identify potential integration issues before deployment.

References

1. Industrial Digital Twin Association. (2023). “Specification of the Asset Administration Shell – Part 1: Metamodel”
2. Object Management Group. (2017). “Unified Modeling Language (UML) Version 2.5.1”
3. World Wide Web Consortium. (2014). “RDF 1.1 Concepts and Abstract Syntax”
4. International Organization for Standardization. (2020). “ISO/IEC 11179-3 Information technology – Metadata registries (MDR) – Part 3: Registry metamodel and basic attributes”