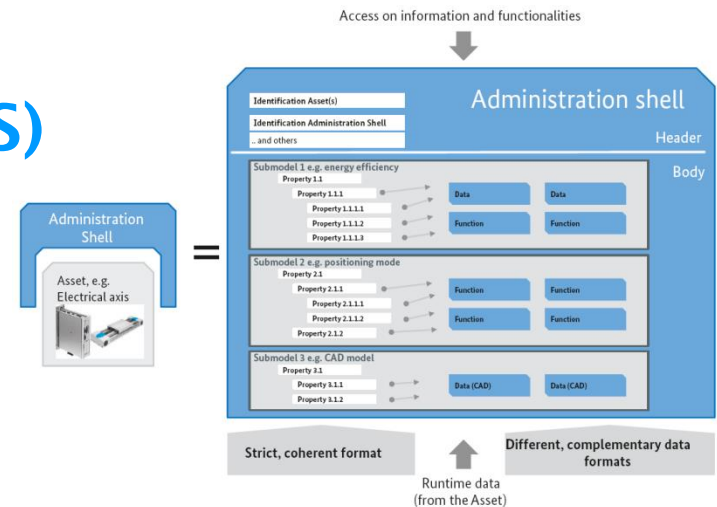
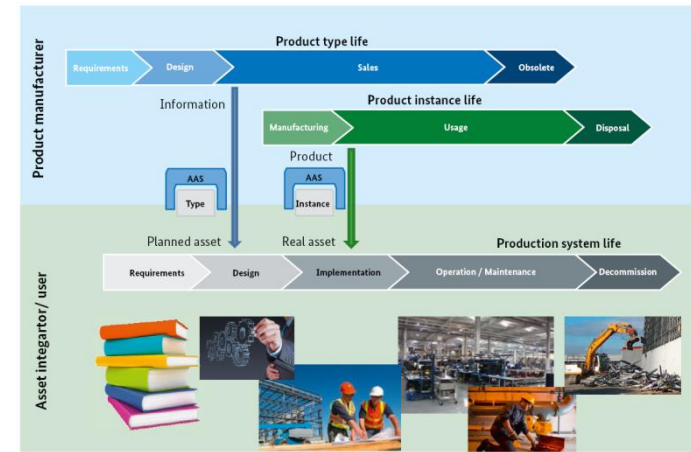


Filling an Asset Administration Shell

Arndt Lüder

Otto-v.-Guericke Universität Magdeburg, Germany

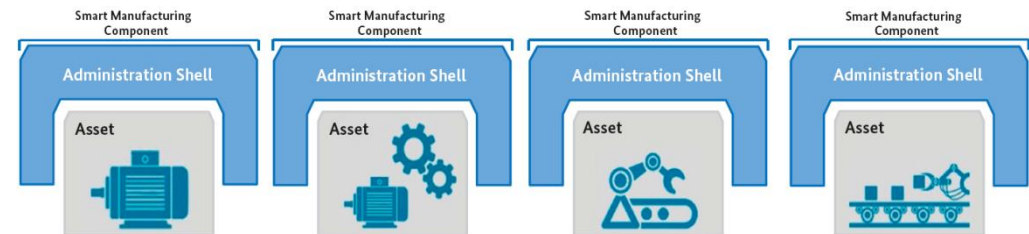
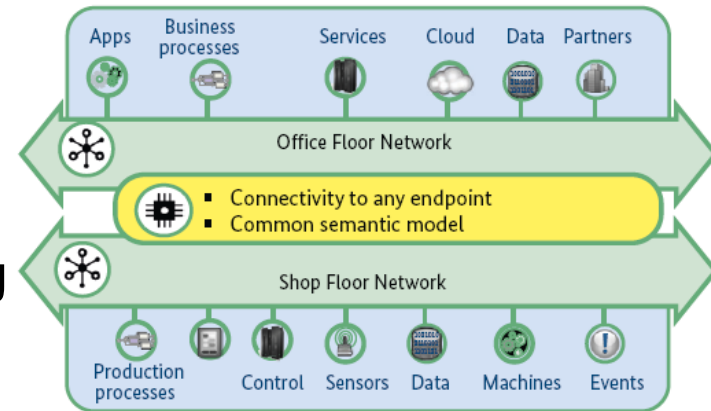
- **CPPS are key concept for modern automation systems**
 - Combination of physical production process elements and their control with increasing intelligence and self-* capabilities
 - Defined within RAMI 4.0
- **Intelligence is covered by Industry 4.0 Asset Administration Shell (AAS)**
 - Providing identification and communication capabilities
 - Combining engineering information and runtime data
 - Giving access to technical functions of the CPPS



Source: The Structure of the Administration Shell: TRILATERAL PERSPECTIVES from France, Italy and Germany, 2018

■ Asset Administration Shell shall be usable

- On different levels of the production system hierarchy from simple devices up to complex value adding systems
- During the complete life cycle of the considered asset (including engineering phase)
- In various use cases of information processing
- To ensure interoperability of all information processing cases

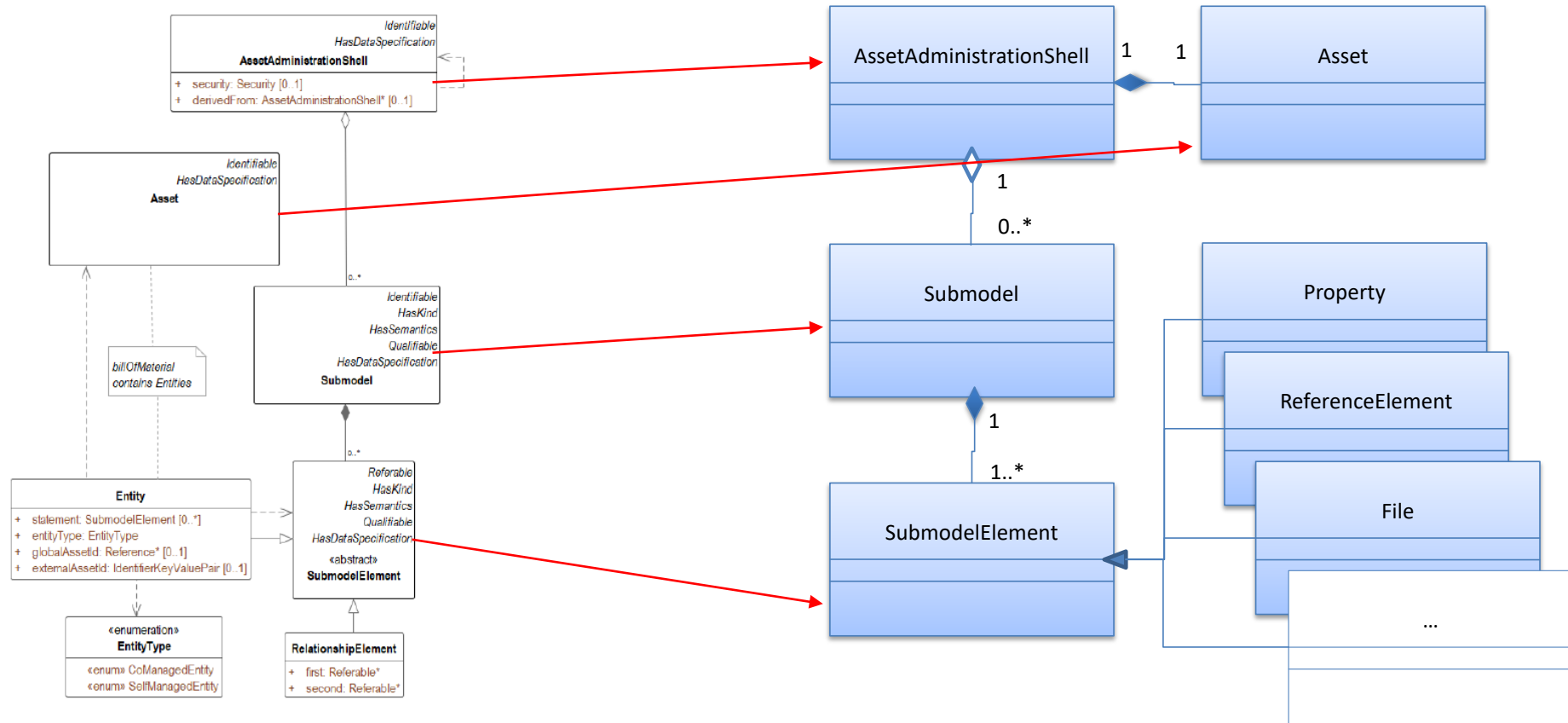


■ Open issue:

- Which information shall be covered by an AAS, how shall it be structured, and how shall it flow?

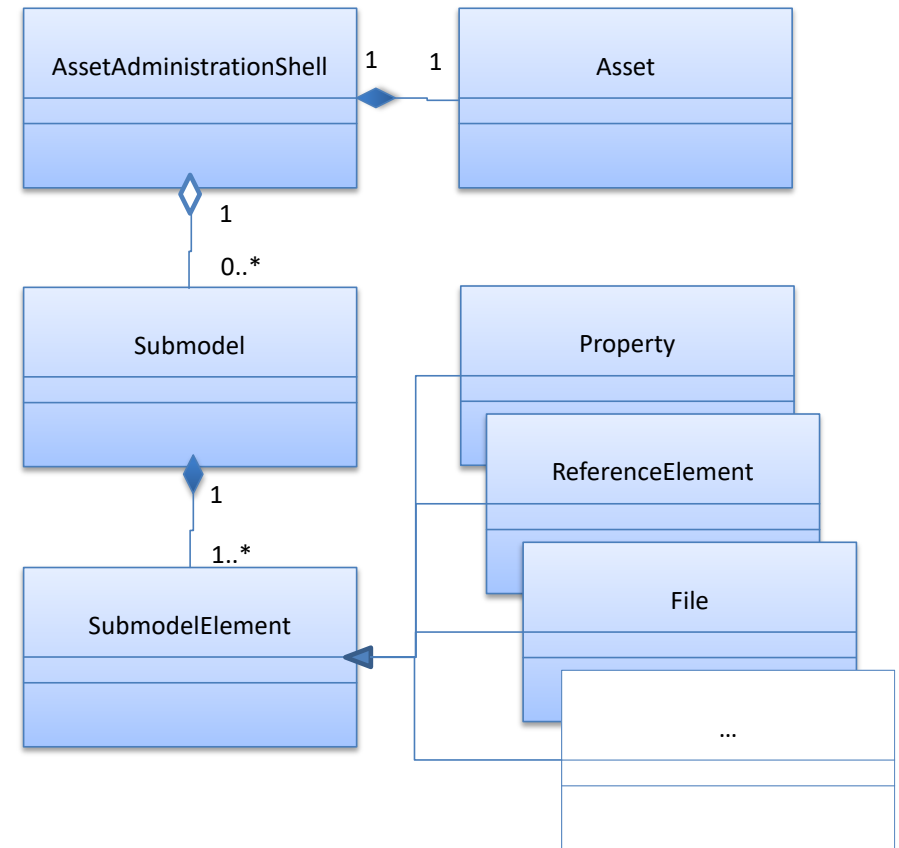
Source: The Structure of the Administration Shell: TRILATERAL PERSPECTIVES from France, Italy and Germany, 2018

- **Standardization of AAS structure and behaviour**
 - AAS structure has been defined within a meta model



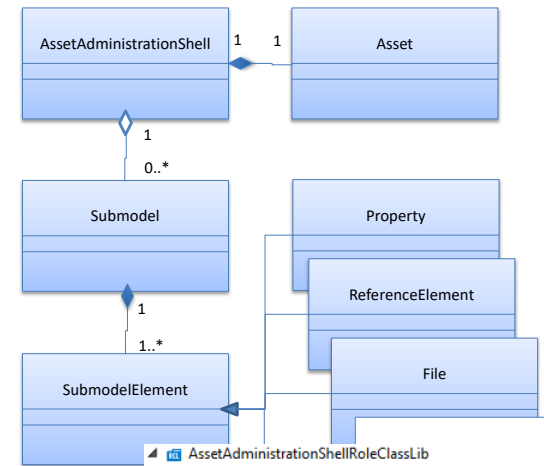
■ Standardization of AAS structure and behaviour

- one asset data object representing the identification information of the linked asset
- Set of submodels representing information items that are technically separated from each other but together form a consistent model of the asset of interest
- Each submodel may contain properties, references, files, and more in a structured and semantically unique way



■ AutomationML serialization of AAS

- Mapping rules between I4.0AAS models and AutomationML models
- Define a set of role classes, interface classes, and attributes representing the I4.0AAS meta model structure elements
 - I4.0 AAS submodels represented by appropriate AML IE with the role *Submodel*
 - Submodel Element represented by IE with appropriate role and an internal structure matching type of Submodel Element



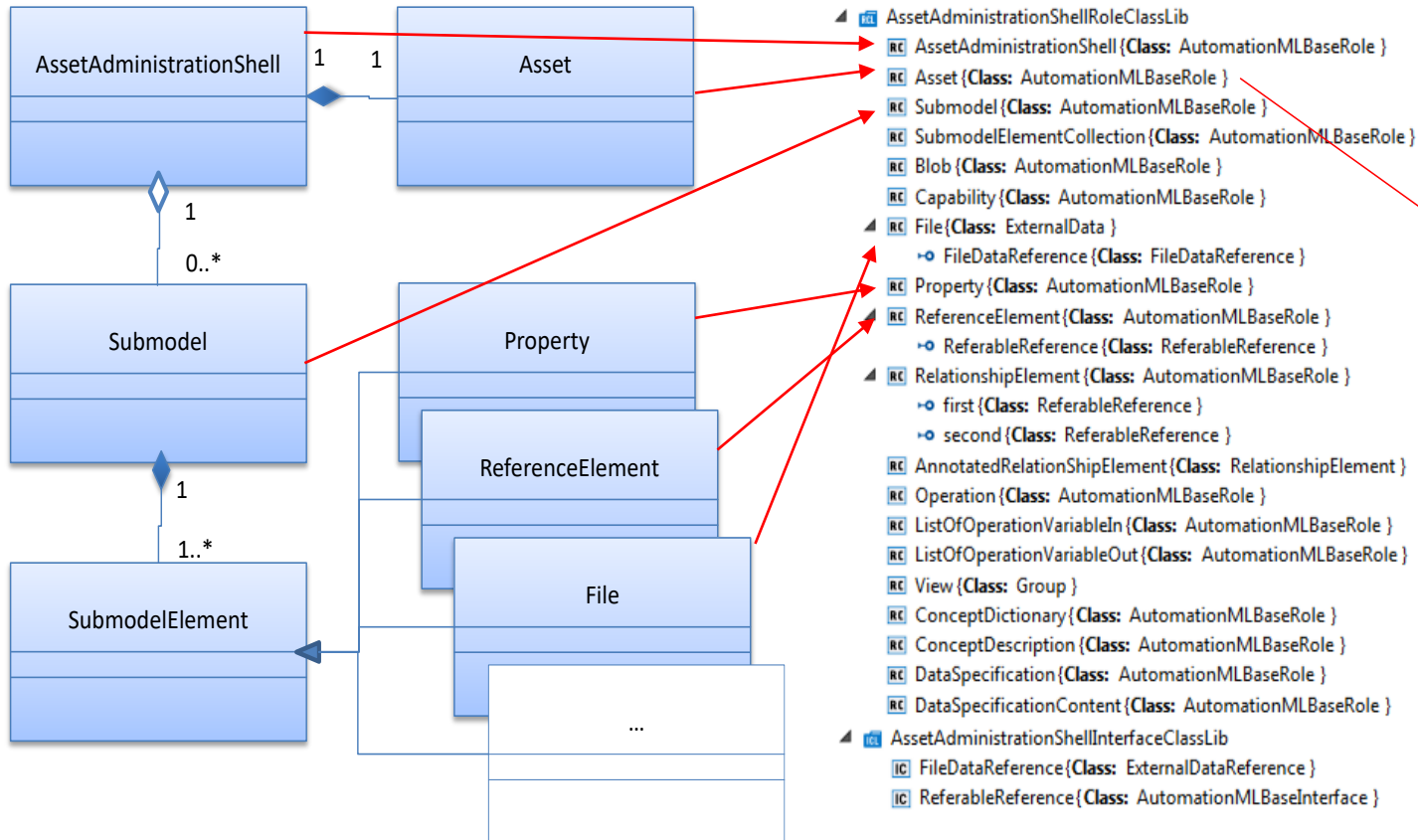
```

AssetAdministrationShellRoleClassLib
  AssetAdministrationShell {Class: AutomationMLBaseRole }
  Asset {Class: AutomationMLBaseRole }
  Submodel {Class: AutomationMLBaseRole }
  SubmodelElementCollection {Class: AutomationMLBaseRole }
  Blob {Class: AutomationMLBaseRole }
  Capability {Class: AutomationMLBaseRole }
  File {Class: ExternalData }
    FileDataReference {Class: FileDataReference }
  Property {Class: AutomationMLBaseRole }
  ReferenceElement {Class: AutomationMLBaseRole }
    ReferableReference {Class: ReferableReference }
  RelationshipElement {Class: AutomationMLBaseRole }
    first {Class: ReferableReference }
    second {Class: ReferableReference }
  AnnotatedRelationshipElement {Class: RelationshipElement }
  Operation {Class: AutomationMLBaseRole }
  ListOfOperationVariableIn {Class: AutomationMLBaseRole }
  ListOfOperationVariableOut {Class: AutomationMLBaseRole }
  View {Class: Group }
  ConceptDictionary {Class: AutomationMLBaseRole }
  ConceptDescription {Class: AutomationMLBaseRole }
  DataSpecification {Class: AutomationMLBaseRole }
  DataSpecificationContent {Class: AutomationMLBaseRole }

AssetAdministrationShellInterfaceClassLib
  FileDataReference {Class: ExternalDataReference }
  ReferableReference {Class: AutomationMLBaseInterface }
  
```

Name	Data Type
administration	Empty
revision	xs:string
version	xs:string
category	xs:string
description	xs:string
aml-lang=EN	Empty
aml-lang=DE	Empty
hasDataSpecification	xs:string
identification	Empty
id	xs:string
idType	xs:string
idShort	xs:string
kind	xs:string

AutomationML serialization of AAS

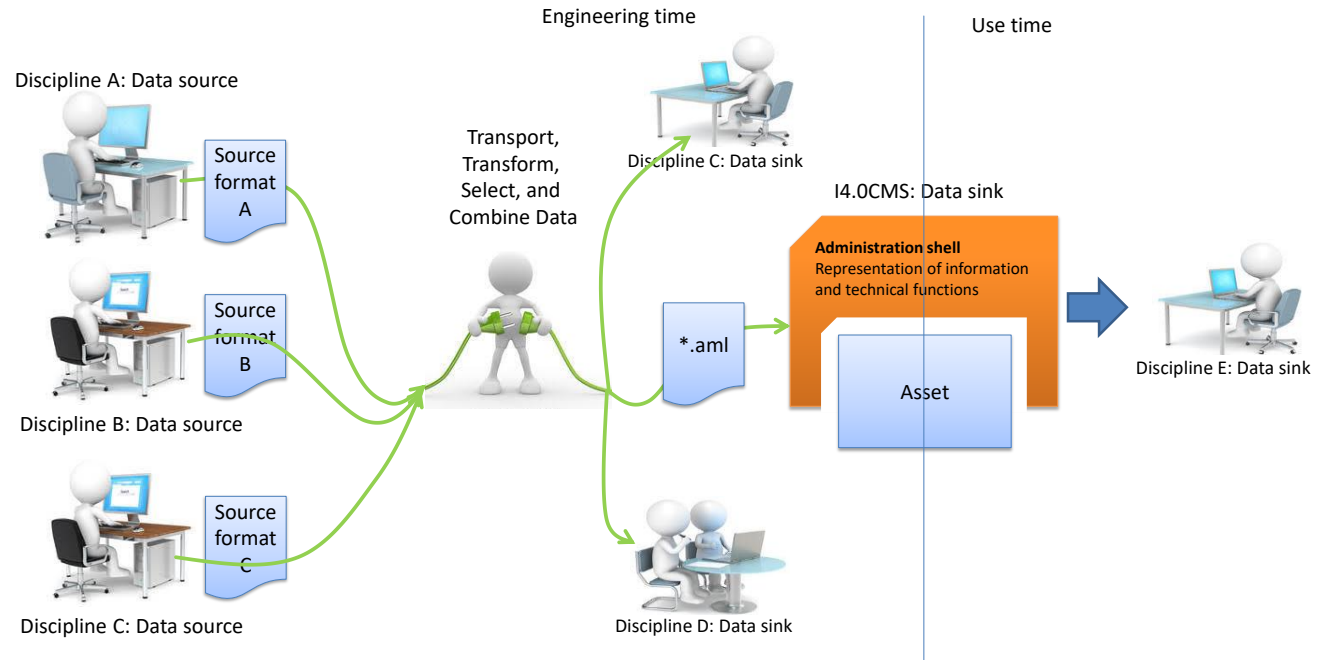


Name	DataType
administration	Empty
revision	xs:string
version	xs:string
category	xs:string
description	xs:string
aml-lang=EN	Empty
aml-lang=DE	Empty
hasDataSpecification	xs:string
identification	Empty
id	xs:string
idType	xs:string
idShort	xs:string
kind	xs:string

- **Open issue: What are the sub-models to be selected?**
 - **Currently there are first submodels under development**
 - **Nameplate**
 - Identifying, descriptive and indicating information about an asset
 - Contains for example manufacturer name, model type or serial number
 - **Technical data**
 - Contains four data sets covering
 - General information (minimal information about the provider of the industrial equipment and the equipment itself)
 - Product classification
 - Technical properties detailing on technical data
 - Further information such as textual statements by the manufacturer and date of validity
 - **BUT is this sufficient?**
-

■ CPPS engineering is characterized as

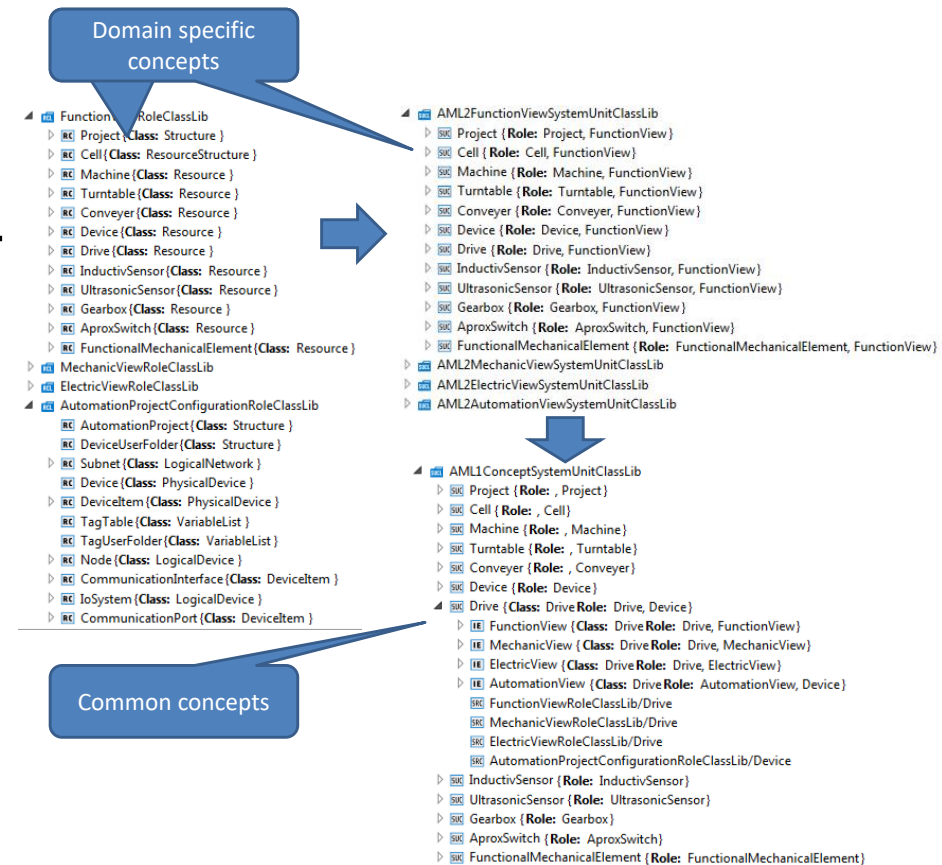
- Network of interrelated engineering-discipline-specific design decisions
- Based on engineering tools and data exchange structures
- Depending on exchange of information
- Applying wide variety of data models, data formats, and modelling means (Model-Driven Engineering)
- Can be considered as network of DSLs



- **Explicit utilization of domain specific languages (DSL) and their dependencies within engineering networks**

- Each DSL is mapped to one AML dialect as role, interface and system unit class system
- System of DSLs is mapped to a use case specific common AML system unit class system utilizing the AML basic dialects

- **Exploitation within an enhanced data logistics with appropriate adapters for model transformation**



Representation of global DSL based on combination of IEs of local DSLs with attribute based interlinking (use of refSemantic)

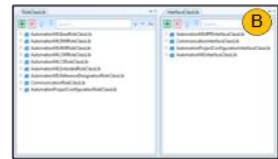
Representation of local DSL based on appropriate role classes with characterizing attributes

```

IE PS_SubS_UMC_VAP_MM_CD1_D1 {Class: Drive Role: Drive, Device, AssetAdministrationShell}
└─ IE FunctionView {Role: Drive, FunctionView, Submodel}
    ├── IE FunctionViewID {Role: Property}
    ├── IE Description {Role: Property}
    ├── IE Manufacturer name {Role: Property}
    ├── IE Manufacturer_part_number {Role: Property}
    ├── IE Functional_description {Role: Property}
    ├── IE Rated_voltage {Role: Property}
    ├── IE RatedPower {Role: Property}
    ├── IE Rated_Current {Role: Property}
    ├── SRC HierarchyViewRoleClassLib/FunctionView
    ├── SRC AssetAdministrationShellRoleClassLib/Submodel
    └── RR FunctionViewRoleClassLib/Drive
    ├── IE MechanicView {Role: Drive, MechanicView, Submodel}
    ├── IE ElectricView {Role: Drive, ElectricView, Submodel}
    └── IE AutomationView {Role: AutomationView, Device, Submodel}
        ├── SRC MechanicViewRoleClassLib/Drive
        ├── SRC ElectricViewRoleClassLib/Drive
        ├── SRC AutomationProjectConfigurationRoleClassLib/Device
        ├── SRC AssetAdministrationShellRoleClassLib/AssetAdministrationShell
        └── RR FunctionViewRoleClassLib/Drive
  
```

idShort	FunctionViewID
category	CONSTANT
kind	Type
semanticId	(Submodel)(local)[URI] FunctionViewID
dataSpecification	
value	PS_SubS_UMC_VAP_MM_CD1_D1

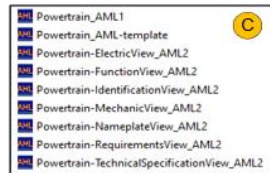
idShort	Rated_voltage
category	PARAMETER
kind	Type
semanticId	(Submodel)(local)[URI] FunctionViewID
value	10



Base-classes.aml



Common-Concept.yml



Common-Concept.yml



TransformerConfig.yml

Generator
Step0

Transformer
Step1

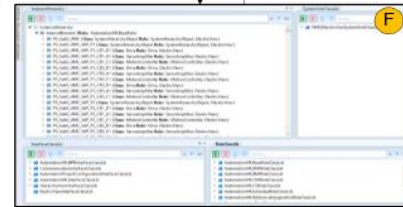
AML1
Transformer
Step2

Integrator
A3

Tracer
A4

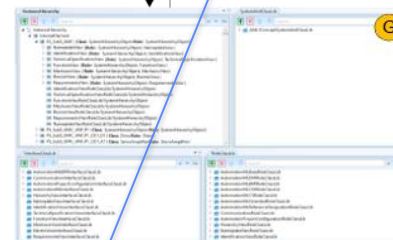


ToolData.*

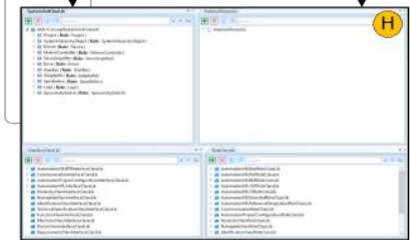


xViewAML2Data.aml

PostProcessor
Step1*



xViewAML1Data.aml



ProjectAML1Data.aml

Step 0: Generates required Role and System Unit Class libraries to build local and global DSL-based data models

Step 1: Translate tool artifacts to engineering artifacts based on AML-based local DSL that fits to the local DSL of the source tool

Step 2: Maps the engineering artifacts to the AML-based structure of the global DSL

Step 4: Fills linked attributes by addressing dependencies between attributes of different domains

Step 3: Integrate different engineering artifacts in the overall AML-based project file

Which I4.0 asset administration shell (AAS) submodels shall be used?

- Mapping of each DSL to one representing AML dialect
- Mapping of these AML dialects to views within the overall AML model
- Each View is equivalent to a submodel

➔ Submodels shall be based identified by DSLs within engineering tools and DSLs used for data exchange between engineering tools

Submodells with their submodel elements

```
IE PS_SubS_UMc_VAP_MM_CD1_D1 {Class: Drive Role: Drive, Device}
  IE FunctionView {Role: Drive, FunctionView}
  IE MechanicView {Role: Drive, MechanicView}
  IE ElectricView {Role: Drive, ElectricView}
  IE AutomationView {Role: AutomationView, Device}
    SRC AutomationProjectConfigurationRoleClassLib/Device
    RR HierarchyViewRoleClassLib/AutomationView
    SRC MechanicViewRoleClassLib/AutomationView
    SRC ElectricViewRoleClassLib/AutomationView
    SRC AutomationProjectConfigurationRoleClassLib/AutomationView
    RR FunctionViewRoleClassLib/AutomationView
```

ElectricViewID	PS_SubS_UMc_VAP_MM_CD1_D1
Drive Profile	
Rated current	220
Rated voltage	10
Axle diameter	200
Appl. output speed	200
Nominal voltage	24
Nominal current	0

FunctionViewID	PS_SubS_UMc_VAP_MM_CD1_D1
Description	Drive
Manufacturer name	Some Drive Provider
Manufacturer part number	ABC 123 DEF 456.78 GH
Functional description	DC motor
Rated voltage	10
Rated power	22
Rated current	220
Max. output speed	1000

■ Transformation of overall AML dialect to I4.0AAS serialization required

- Implemented by a special adapter for the engineering data logistics
- Adapter shall
 - add necessary role and interface classes
 - translate AutomationML attributes to *IEs* with the role Property with a special attribute structure by respecting naming and structuring rules

Submodel elements expressed as IEs with standardized attributes

idShort	FunctionViewID
category	CONSTANT
kind	Type
semanticId	(Submodel)(local)[URI] FunctionViewID
dataSpecification	
value	PS_SubS_UMC_VAP_MM_CD1_D1

idShort	Rated_voltage
category	PARAMETER
kind	Type
semanticId	(Submodel)(local)[URI] FunctionViewID
value	10

Benefits:

- (1) Solution can be introduced incrementally within an EO without loss of previous efforts/investments.
- (2) Solution facilitates different testing methods and approaches following different functionality enrichment scenarios (tool artifacts, data within them, data propagation needs).
- (3) Utilization of solution is cost-efficient
 - no programming effort
 - but configuration effort based on knowledge about the local DSLs to be connected
- (4) Solution provides foundations for defining different migration strategies towards a full implementation of data logistics.

Limitations:

- (1) Solution requires a unique identification of all objects along the EO tool chains.