This regular paper was presented as part of the main technical program at IEEE ETFA'2011

Information Modeling for Middleware in Automation

Wolfgang Mahnke
ABB Corporate Research
Industrial Software Systems
68526 Ladenburg
wolfgang.mahnke@de.abb.com

Andreas Gössling
Technische Universität Dresden
Institute of Applied Computer Science
01062 Dresden
andreas.goessling@tu-dresden.de

Markus Graube, Leon Urbas
Technische Universität Dresden
Institute of Automation
01062 Dresden
markus.graube@tu-dresden.de
leon.urbas@tu-dresden.de

Abstract

Middleware in automation serves the connection of hardware and software components of applications in automation-like control or human-machine interfaces. To cope with today's flexibility requirements, there is an urgent demand to not only communicate data but to describe the semantics of the distinct elements of the distributed information space. Therefore, middleware in automation needs to provide some means for information modeling. To derive general concepts and recommendations on how to implement information models on middleware in automation, we discuss several relevant standards that provide partial information models. We then identify the fundamental information modeling constructs of these partial models and derive general requirements for the information modeling capabilities of middleware in automation.

1. Introduction

Whenever automation related hardware and software of different vendors has to be connected for some application that goes beyond archiving process values or alarming, an engineer might want to consider utilizing middleware. For best results the middleware not only cares for data handling and exchanging data but also provides information that describes the semantics of the data [5]. To catch the semantics of a simple measurement value one might need to add information about the measurement time, engineering units and the measurement range as well as information about the current state of the device, like status or selected measurement method. While most of this information could be held in some independent data store, it turns out that for practical reasons there is an urgent demand that the middleware itself provides this information in a structured way. Eventually, this would require that any middleware provides some means of information modeling. The resulting solutions can be varied, ranging from a generic model – for example describing common characteristics of devices – to very concrete models, for example defining concrete measured values and configuration options of a specific class of devices.

2. Middleware in Automation Systems

In automated systems (or even more generally in any distributed information processing system) any device or subsystem needs to implement similar communication-related functionalities like search, connect, authorize, adapt, send and receive, etc. One expensive solution would be to have all of the subsystems implementing the same (or at least functionally identical) pieces of software. Very often it is of advantage to implement these common functions in a distinct system, which then provides interfaces to the subsystems. They then utilize the function-in-the-middle instead of implementing them. The possible benefits of such a middleware design pattern are straightforward: Complex applications need to bother only with interfaces that conceal any complicated implementation details of the particular devices. The devices themselves can be kept simple, because the complicated part of communications and persistence are realized in the middleware. This clear separation of functionality has secondary effects for further distributed development, reuse, reduction of complexity, higher flexibility and maintainability.

These advantages, however, come at a price. It is not sufficient only to define how to securely connect, code, and exchange data. At the end of the day it is necessary to define the semantics of the information on the middleware level, too. Fixed semantics of information models turn out to allow for starting off rapidly – however one can observe a "creative erosion" of the semantics during the lifetime of the middleware as new applications and requirements can be met only by stretching the meaning of certain data fields. Therefore, to keep up with changing requirements, middleware in automation needs to provide some means to flexibly describe and cover the context and semantics of an automated system in an abstract way.

978-1-4577-0018-7/11/$26.00 ©2011 IEEE

3. Information Models in Automation

Talking about abstraction directly leads towards the question how to construct generic information models. Attempts have been made to distill models from analyzing existing models and finding a common denominator [1].However, unstructured generative approaches have proven hard to handle in the face of the variety of models in automation. To come to an informed decision about which properties and means of abstraction middleware shall implement, we analyzed several prominent standardized information models. The selection is not exhaustive; however the choice covers the complete life cycle as well as different levels and domains.

3.1. IEC/EN 62424 (CAEX)

CAEX provides a meta-model that provides means to define hierarchical plant models and describe their attributes. To that end, CAEX defines a small set of concepts that are sufficient to describe functional and topological hierarchies as well as the composition of subsystems by aggregating and relating distinct entities. These top-level classes are named basis elements, interfaces and components [3]. This is accompanied by a pragmatic role and library concept. On top of these concepts IEC/EN 62424 defines a model for the information exchange between chemical process engineering and automation system engineering. Figure 1 shows the fundamental structure of the information model as a UML class diagram. Highlighted are process control equipment request class and its components *SignalInterface* and *ProcessConnectionInterface*.

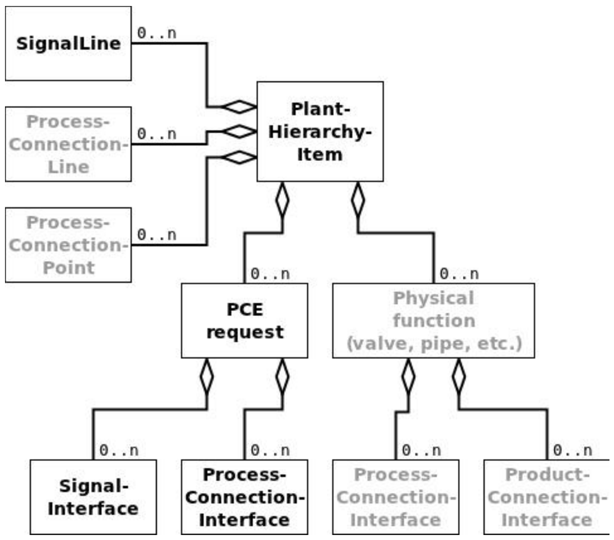
SignalLine
Process-Connection-Line
Process-Connection-Point
Plant-Hierarchy-Item
PCE request
Physical function (valve, pipe, etc.)
Signal-Interface
Process-Connection-Interface
Process-Connection-Interface
Product-Connection-Interface
0..n

Figure 1.Basic Elements of IEC/EN 62424 for modeling Process Control Equipment requests

To summarize, IEC/EN 62424 defines an information model with a number of top-level classes. Instead of the usual strict interface-contracts of object-oriented approaches, CAEX proposes a role concept that foresees gradual refinement and specification by appointment. Checks that guarantee that an element really "fills the position" are, however, outside the scope of CAEX.

3.2. ISO 15926 (Lifecycle Information Exchange)

ISO 15926 sees itself as an ontology that formalizes temporal, spatial and functional aspects of all the "things" that make up a production process. It strives to cover the whole life cycle from engineering to operation. Accordingly, "things" may be actual or possible physical entities like devices as well as abstract concepts like classes or relations. Attributes may be implemented as elementary data types or as references to other entities. Abstract concepts like class, inheritance and multidimensionality are defined as basic entities in their own right (Figure 2).

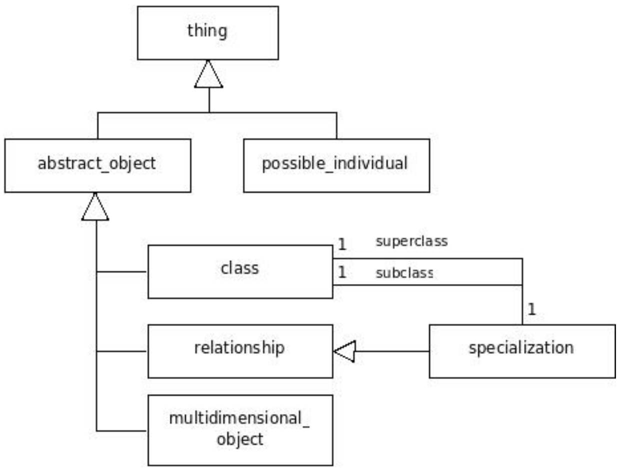

Figure 2. ISO 15926 Basic Model Elements

The two major modeling strategies are subtype/supertype hierarchies and temporal and special composition by relational entities. This leads to a well-differentiated predefined subtype/supertype hierarchy. For example, the abstract class 'pump' is an instance of *class_of_functional_object*, which is derived from *class_of_arranged_individual* and adds the notion of functionality. Any concrete pump is an instance of the type *class_of_inanimate_physical_object* which itself is a subtype of *class_of_functional_object*. To summarize, ISO 15926 defines a hierarchical model with subtypes that are derived from a root element "thing" that carries all the information about identity and the derived abstract basic entities class, relationship and multidimensional object. Attributes are implemented as instances of basic data types or references to other "things". With these basic principles ISO 15926 defines a complex ontology that copes with the changing spatial, temporal and functional relations between all the information that needs to be handled during the life cycle of a plant.

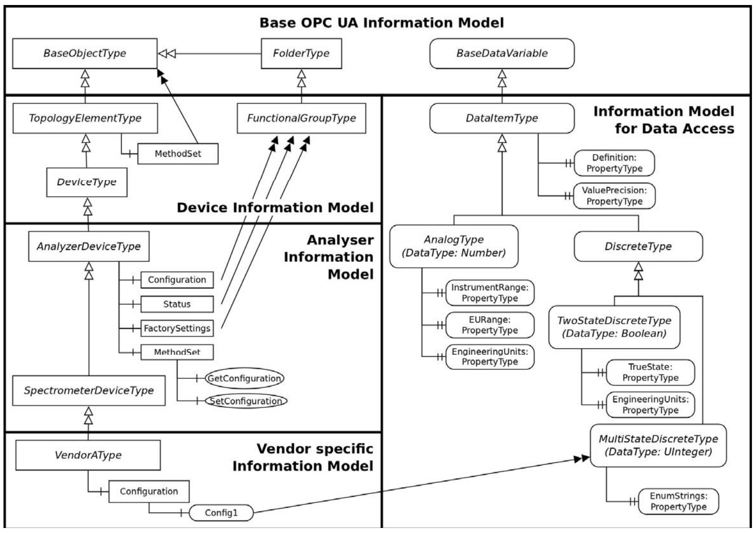
Base OPC UA Information Model
BaseObjectType
FolderType
BaseDataVariable
TopologyElementType
FunctionalGroupType
MethodSet
DeviceType
Device Information Model
DataItemType
Information Model for Data Access
Definition: PropertyType
ValuePrecision: PropertyType
AnalyzerDeviceType
Analyser Information Model
Configuration
Status
FactorySettings
MethodSet
GetConfiguration
SetConfiguration
SpectrometerDeviceType
AnalogType (DataType: Number)
InstrumentRange: PropertyType
EURange: PropertyType
EngineeringUnits: PropertyType
DiscreteType
TwoStateDiscreteType (DataType: Boolean)
TrueState: PropertyType
EngineeringUnits: PropertyType
MultiStateDiscreteType (DataType: UInteger)
EnumStrings: PropertyType
VendorAType
Vendor specific Information Model
Configuration
Config1

Figure 3. Excerpt of the ADI model and its relation to other Information Models

3.3. EDD, FDT, FDI, ADI (Device Integration)

The field of device integration is characterized by several overlapping modeling approaches. Electronic Device Description (EDD) provides means to model the information space of an electronic device. EDD covers not only the exposed structure of parameters but also describes the organization and layout of a possible user interface. Field Device Tool (FDT) provides a complex interface model that covers communication and parameters and defines a framework to integrate binary applications and drivers. Field Device Integration (FDI) combines both approaches towards a generic device model. In contrast to FDI, the models of Analyzer Device Integration (ADI) describe concrete classes and their attributes. Both FDI and ADI utilize the OPC UA Information Model as depicted in Figure 3. The basic OPC UA information model is extended with a data access model, a generic device information model and a special subtype model for the analyzer part. This part may be extended by the vendors to address specific abilities not part of the analyzer model. Type hierarchies and inheritance are the main modeling constructs. Attributes and parameters may be specified by different types of references. Finally, the modeling framework includes method invocation. The models are extensible by inheritance mechanisms.

3.4. IEC/EN 61131-3 (Programmable Logic Controller)

IEC/EN 61131-3 defines an architecture model for programmable logic controllers (PLC). The main elements are configuration, resource, program organization elements (POE) and variables (see Figure 4). A configuration is an automation task specific collection of networked automation resources; typically PLCs. Every resource runs several tasks which control the execution of assigned POEs and may host global variables. POEs are programs (no input/output parameters, instance variables), function blocks (input/output parameter lists, local instance variables) and functions (no local instance variables, output values depend on input values only). IEC/EN 61131-3 defines the syntax of several automation specific programming languages, including the naming of variables and the syntax of literals and defines a set of predefined function blocks and functions that any PLC shall implement. The programming model's definitions could be used as a solid basis for a formal model describing the state of a PLC. Any middleware that supports structured and user defined data types as well as relations for hierarchies and aggregation can describe the variable space of IEC/EN 61131-3 conform PLCs. A recent example is the joint action of PLCopen and the OPC Foundation to define an OPC UA information model [6].

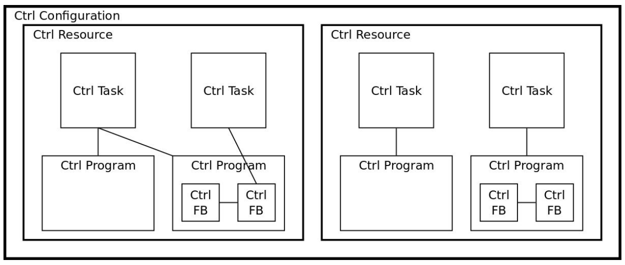

Figure 4.Software Model IEC 61131-3

3.5. IEC/EN 61512 (ISA-88)

IEC/EN 61512 defines a terminology and models for chemical plants including process equipment and control instrumentation as well as procedures and recipes for batch control (see Figure 5 and Figure 6). The process model describes chemical processes as a hierarchy of sections, operations and steps. By means of a recipe, a process is implemented on a physical process cell. Process cells can be identified as part of an enterprisesite-plant hierarchy. They may consist of several units, which are a composition of equipment modules and their associated control modules.

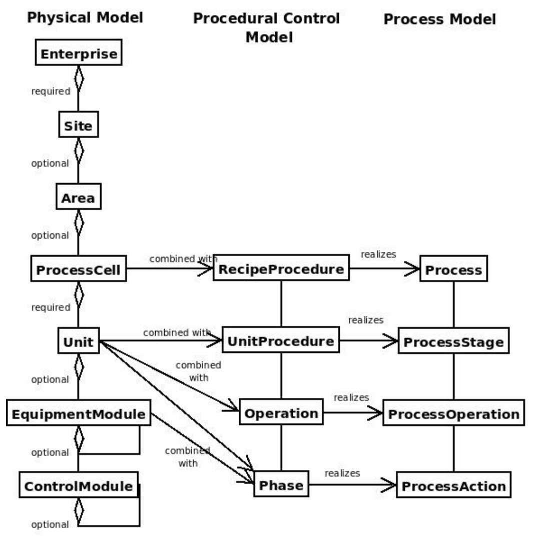

Figure 5. Summary of IEC 61512

IEC/EN 61512 covers these concepts on a rather abstract level. It explicitly does not aim to propose an architecture for a batch control system. Accordingly all of the entities, aggregations and attributes are defined without any specification for internal data types. Instead of that, the standard defines structures and formats for tables to exchange data between different systems. The most frequently used data types are extensible EnumerationSets (values from 0 to 99 are specified individually for any entity), Date and Time, Coordinates, and Strings. Attributes of parameters may follow IEC/EN 61131-3.

All in all, IEC/EN 61512 defines a relational data model with predefined entities, attributes and relations. While the structure of the model is fixed, the exchange model may be extended by user-defined values for the EnumerationSets.

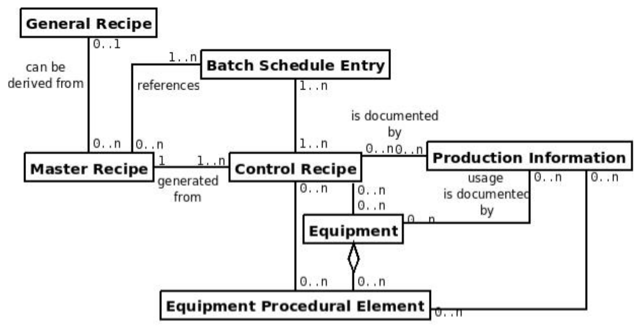

Figure 6. Overview Model IEC 61512

3.6. IEC/EN 62264 (ISA-95)

This standard provides guidance for the integration of shop floor information with the enterprise information processing systems. The information model covers hierarchies, information flow and objects (Figure 7). Hierarchies are used to allocate equipment within the automation pyramid; information flow models define the exchange between production-related functions. A set of predefined objects are formalized with UML (Part 1) and denote the processes that are connected by means of information flows. Users may adapt this fixed object model to their needs by adding properties. Part 2 defines the attributes of the objects without specifying the data types of the implementation. Part 3 finally denotes the functions of the manufacturing execution system level for manufacturing. Basic domains are production, maintenance, quality assurance and inventory.

IEC/EN 62264 provides an object-oriented model of production processes by means of UML. Users are asked to adapt this model by specifying properties of predefined classes. Central means of modeling are aggregation and association, inheritance is mentioned but not used. Data types are thought to be implementation specific.

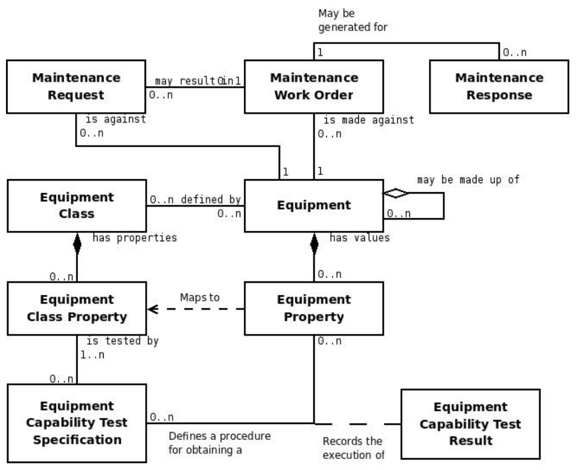

Figure 7. Equipment Model of IEC 62264

3.7. IEC/EN 61850 and IEC/EN 61400

IEC/EN 61850 defines a basic information model which is used to describe specific models for substations (Part 7-4), hydro-power generation (Part7-410), decentralized power generation (Part 7-420) or wind power generators (IEC/EN 61400), e.g. the standard models the world as a collection of servers that hold logical devices. Devices consist of logical nodes that host a hierarchy of data objects which aggregate typed data attributes (see Figure 8). The specific models already provide a great number of domain specific logical nodes which may be extended by users. Predefined data objects are re-used across nodes. Following the object-oriented paradigm these objects already carry semantics. In addition, the information model is accompanied by an information exchange model that covers several aspects of communication. Via substation configuration language (SCL) optional attributes may be added. The information model uses data types and aggregation, but not inheritance. Extended models may add vendor specific node and object classes.

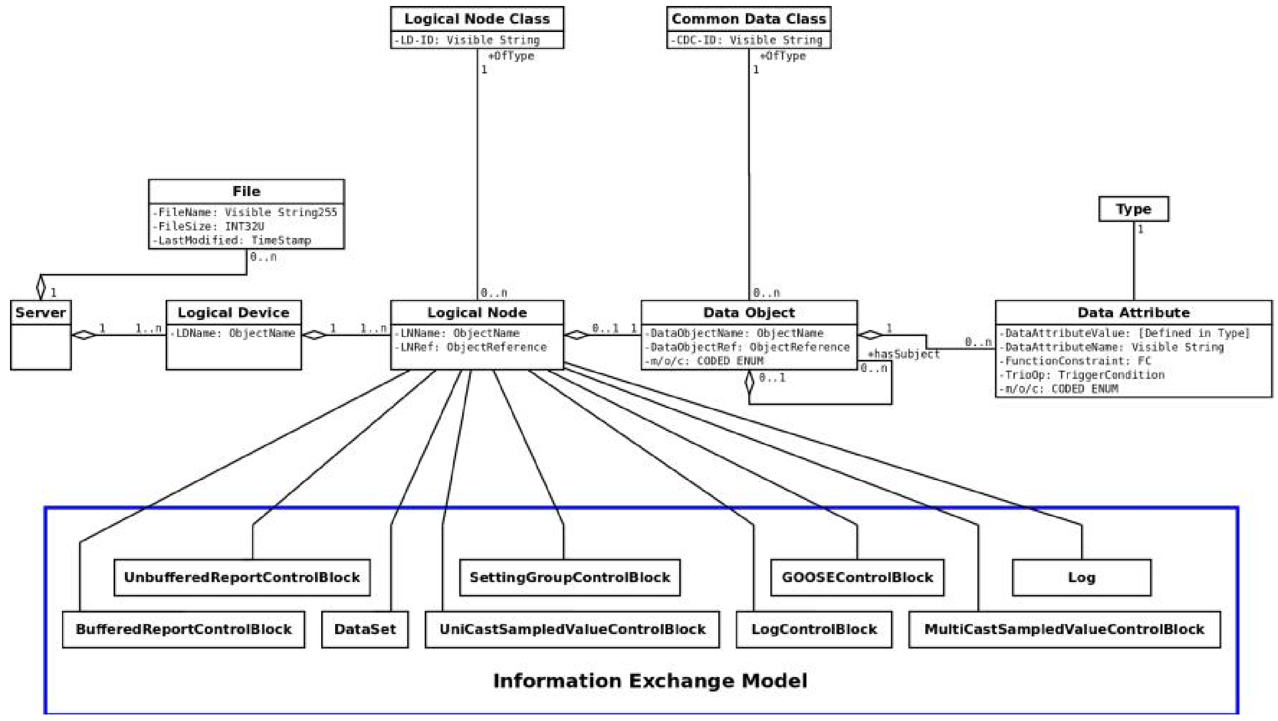

Figure 8. IEC 61850 Information Model

3.8. IEC/EN 61970 and IEC/EN 61698

IEC/EN 61970 defines a model for energy transmission, IEC/EN 61968 for energy distribution. The first standard introduces a common information model (CIM) which is then extended by the second. The CIM is formally described with basic UML concepts (See Figure 9). Packages are used to structure an object-oriented class model with inheritance, attributes and relations. Figure 10 depicts the model of a steam turbine with this standard. The root of the object inheritance tree is *Core::IdentifiedObject*, which carries all the information that is necessary to uniquely identify the steam turbine (quite similar to "thing" in ISO 15926), followed by *Core::PowerSystemResource*, *PrimeMover*, and eventually *SteamTurbine*. All of the necessary attributes and relations are added by subtyping.

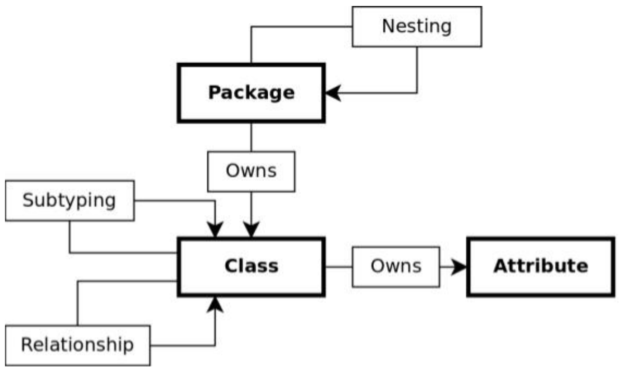

Figure 9. Base concepts of the CIM

CIM applies the UML-Concept of a class including attributes and inheritance as well as relations with cardinal restrictions. Pragmatic grouping aspects are handled by a hierarchical package concept. CIM is intended to be extensible on all levels – classes, attributes, and packages.

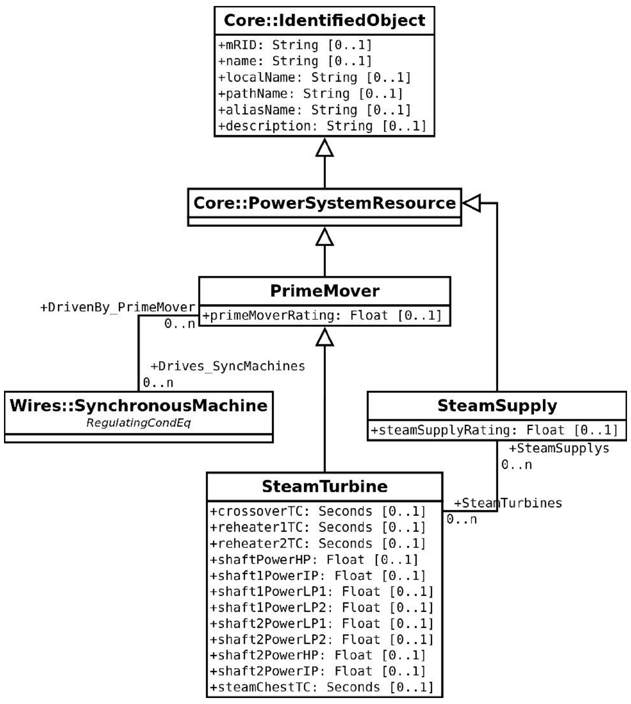
Core::IdentifiedObject
+mRID: String [0..1]
+name: String [0..1]
+localName: String [0..1]
+pathName: String [0..1]
+aliasName: String [0..1]
+description: String [0..1]
Core::PowerSystemResource
PrimeMover
+primeMoverRating: Float [0..1]
+DrivenBy_PrimeMover
0..n
+Drives_SyncMachines
0..n
Wires::SynchronousMachine
RegulatingCondEq
SteamSupply
+steamSupplyRating: Float [0..1]
+SteamSupplys
0..n
SteamTurbine
+crossoverTC: Seconds [0..1]
+reheater1TC: Seconds [0..1]
+reheater2TC: Seconds [0..1]
+shaftPowerHP: Float [0..1]
+shaft1PowerIP: Float [0..1]
+shaft1PowerLP1: Float [0..1]
+shaft1PowerLP2: Float [0..1]
+shaft2PowerLP1: Float [0..1]
+shaft2PowerLP2: Float [0..1]
+shaft2PowerHP: Float [0..1]
+shaft2PowerIP: Float [0..1]
+steamChestTC: Seconds [0..1]
+SteamTurbines
0..n

Figure 10. Example of a Steam Turbine

4. Information Modeling Concepts

The short presentation of the various standards in the previous sections shows that the structure of the information models is quite different. For simple models it seems to be sufficient to rely on simple structuring principles. Compositional relations (*is-part-of*) are expressed by hierarchies or aggregations. Named variables are used to exchange data. The specification of the data type might be part of the implementation. Basic data types like integer or double are used as well as complex ones (structures). Some information models allow extension of the model by user-defined data types or include method invocation. References are used to depict further relations between entities. Some information models explicitly allow weak references (similar to a WWW link).

More complex information models introduce classes as a semantically loaded composition of attributes and methods. The class concept can include sub/supertyping-relations by means of inheritance, either internally only or for user added extensions.

Table 1 collects all of the requirements for the information modeling capabilities of any middleware that shall implement the standards modeling constructs directly. Obviously, hierarchies, aggregation and variables are required by all information models. Functions and object-oriented modeling aspects are required only partially. Utilization of a class concept does not automatically include inheritance. Several standards leave the definition of concrete data types to the implementation.

	Hierarchy	Aggregation	Variables	Functions	References	Classes	Methods	Inheritance	Data Types
CAEX	X	X	XML	-	X	-	-	-	XML
ISO 15926	X	X	X	-	X	X	-	X	X
FDI/ADI	X	X	X	X	X	X	X	X	X
IEC 61131-3	X	X	X	?	-	X	?	-	X
ISA-88	X	X	X	-	X	-	-	-	-
ISA-95	X	X	X	X	X	X	-	X	-
IEC 61850	X	X	X	-	-	Partly	-	-	X
CIM	X	X	X	-	X	X	X	X	X

Table 1. Information modeling concepts as applied by the analyzed standards

This analysis would not be complete without looking at the qualitative characteristics of the reviewed standards. These include i) the extensibility of the basic structure and modeling constructs by vendors or end-users, ii) the fundamental modeling philosophy (meta, relational, object-oriented) where meta denotes a model that is intended to define another model and iii) the level of concreteness which may range from meta-models (how to describe a model?) or generic models (how to describe a device?) to concrete, domain-specific models, as shown in Table 2.

	Extensibility	Philosophy	Concreteness
CAEX	Roles	meta	--
ISO 15926	Community Process / Adapter & Facades	Ontology	++
FDI/ADI	Subclasses with additional attributes and methods	OO	+ / ++
IEC 61131-3	Definition of POEs and user defined data types	Function blocks	+
ISA-88	User defined values	relational	++
ISA-95	User defined property values, implementation specific data types	OO	+
IEC 61850	User defined logical nodes and objects, SCL- adaptable optional properties	Mostly OO	++
CIM	Subtyping with inheritance. Add and modify attributes, methods	OO	++

Table 2. Qualitative attributes of analyzed standards

For practical reasons, reflection (the ability to query the model structure) and granularity of access control are important aspects for the implementation of information models in middleware in automation. These aspects are not part of the observed information modeling standards. Nevertheless it is crucial in terms of functionality and performance of the middleware to i) restrict access to authorized users and ii) maximize the utilization of the communication bandwidth by selection of single attribute of interest (instead of complete objects) and grouping of cross-object attributes for bulk transfer.

5. Information Modeling in Middleware

In this section two middleware standards are considered as examples: OPC UA and IEC 61850. Both are relatively new standards developed in this millennium to meet particular communication needs of their industry branches. Both standards currently show a high interest in the industry due to them providing interoperable communication stacks as well as complex information modeling capabilities. We compare their capabilities for information modeling to the requirements of the information models introduced in the previous sections. This gives an overview of which information models can easily be supported by the different middleware approaches, without misusing concepts.

IEC 61850 was introduced as information model in a previous section. In addition to the information modeling capabilities, IEC 61850 provides communication mechanisms and therefore general middleware characteristics [4]. Communication can be via MMS (Manufacturing Messaging Specification – Part 8-1) on top of TCP/IP, or using GOOSE (Generic Object Oriented Substation Events – also Part 8-1) or Sampled Values (Part 9-2) directly on top of Ethernet. IEC 61850 allows fine and coarse granular access to data and mechanisms to exploit the address space. Using SCL, the full configuration of a server can be made available. The relation between standardized or user-defined logical node classes is done by naming rules for instances and namespaces, the structure of the logical node classes can only be exploited using the SCL.

OPC UA provides information modeling and communication capabilities and therefore also middleware characteristics [2]. FDI and ADI introduced in previous sections base on the OPC UA information modeling capabilities. The communication of OPC UA uses technology mapping to web services or to a specialized TCP/IP based protocol. Access to the data is service-oriented, i.e. many variables can be accessed with one request. Meta data are accessed the same way as normal data and all meta data can be exploited. However, currently there is no standardized exchange format available like the SCL in case of IEC 61850. In Table 3 the modeling concepts of the previous section are compared to the capabilities of both middleware standards. As can be seen, OPC UA supports all concepts. However, limitations can be made on details. The semantic of multiple inheritance is not defined by OPC UA. This is because different semantics for multiple inheritance shall be applicable in OPC UA with the drawback that no clear semantic is defined.

	IEC 61850	OPC UA
Hierarchy	X	X
Aggregation	X	X
Variables	X	X
Functions	-	X
References	-	X
Classes	Partly	X
Methods	-	X
Inheritance	-	X
Data Types	X	X

Table 3. Qualitative attributes of analyzed standards

IEC 61850 does not support all modeling concepts, functions or methods, inheritance or references are missing. As IEC 61850 is built for simple structured domain-specific models, those concepts are not needed. Therefore IEC 61850 cannot simply map the ISA 95 information model, but is a suitable mechanism for information exchange for a multitude of domains. By providing the SCL the IEC 61850 already provides mechanisms for engineering and configuration.

With its modeling concepts OPC UA can map all introduced information models. However, OPC UA does not provide any constructs to address the life-cycle; its services and models have a clear focus on the operation phase of an application. There are no adequate formats to configure the server at once or to provide an offline client with a server configuration image. The granularity of accessing data is to a certain degree defined by the model and only partial configurable (e.g. when browsing the address space).

6. Conclusion and Discussion

In this paper a classification of information models for automation is presented, and the applicability of two existing middleware approaches to expose those models discussed. Comparing both approaches, the information modeling capabilities of OPC UA are more powerful and provide more options. On the other hand, IEC 61850 already addresses the engineering phase of an application using the SCL. This can only be a brief overview of existing standards for information modeling in the automation domain. What should be clear after this outlook is that despite standardization, a lot of heterogeneities are to be found in the domain. To overcome these gaps, methods to preserve semantic information over different industries and abstraction levels need to be applied.

References

[1] K. Gnauck; A. Gössling, R. Frenzel, M. Wollschlaeger, D. Schulz, *Development of a Universal Model for Description of Intelligent Field Devices through the Life-Cycle.*ETFA 2009, pp. 1-8, 2009.

[2] W. Mahnke, S.-H.Leitner, M. Damm, *OPC Unified Architecture*, Springer, 2009

[3] G. Mayr, R. Drath, *IEC PAS 62424 - Grafische Darstellung PLT-Aufgaben und Datenaustausch zu Engineering-Systemen.* atp 49 (5), pp. 22-29.

[4] ABB Review Special Report IEC 61850, August 2010, ABB Group R&D and Technology

[5] VDI/VDE guideline 2657, *Middleware in Industrial Automation*, Part I, BeuthVerlag, Berlin, 2011.

[6] PLCOpen, OPC Foundation, OPC UA Information Model for IEC 61131-3, Release 1.00, 2010