Semantic integration by means of a graphical OPC Unified Architecture (OPC-UA) information model designer for Manufacturing Execution Systems

M. Schleipen¹, O. Sauer¹, J. Wang¹

¹ Fraunhofer IOSB, Fraunhoferstr.1, 76131 Karlsruhe, Germany

Abstract

Today, normally MES (Manufacturing execution systems) are isolated applications. Tomorrow, they have to be linked to each other. The prototypical engineering framework based on OPC-UA (OLE for Process control- Unified architecture) processes data in the standardized XML-based format CAEX (Computer Aided Engineering Exchange) describing production plants. It configures MES and generates corresponding process visualization. OPC-UA is a standard which accomplishes to support process communication in a structured way with an underlying user-defined information model. In this contribution, a graphical OPC-UA information model editor is presented which supports CAEX import and OPC-UA export, both in XML. Thus, MES can take advantage of the skills of OPC-UA servers and CAEX data can be managed online. The mentioned concepts were tested with an application example.

Keywords:
Planning, modeling, ontology

1 INTRODUCTION

Today, continuous change and re-planning make the engineering a special challenge for MES [1], [2]. MES have to know their environment to operate without errors. In this case environment means the supervised plant with its signals, their meanings, internal links between plant components, and their graphical representations. The OPC Unified Architecture suits to build up this ‘world model’. OPC-UA represents a modern communication interface between MES and the underlying production. It allows MES to embed received and sent data in an object-oriented description of the real world. This model is permitted by so called information models within only one OPC-UA-Server. These information models have to be created during the planning of a new plant and have to be adapted during re-planning to guarantee the correct assignment, identification and interpretation during operation.
The creation of such an information model can become very time-consuming due to the complexity of the production plant to supervise. Hence, the tool described in this contribution tries to support the planner in doing so. This is realized by means of a graphical tool. Therefore, this tool can also act as help for communication during the plant planning process.

In addition, an import of already existing data in the standardized data format CAEX (Computer Aided engineering exchange) can reduce complexity. The definition of the independent XML-based CAEX data exchange format has been taken down in the standard IEC62424. CAEX is a semi-formal description language, which is based on XML. It contains an XML-meta model for describing the setup and structure of plant data. The format takes account of the problem that the standardization of the tools available on the market only makes sense to a certain degree, if it makes sense at all. First and foremost, the format supports library concepts and object-oriented approaches. It is possible to integrate libraries from users and suppliers as well as project libraries. In addition, both a top-down and a bottom-up system design is supported. The technical innovation of this approach is the syntactic and semantic unification of the considered data. This allows decoupling of the required configuration algorithms from the data sources. [3]

The meta model of CAEX can be interpreted similar to ontology concepts. The Roles within CAEX represent ontology concepts whereas Instances and System Units in CAEX are concrete implementation and in this respect ontology instances. Ontology properties can be represented by CAEX Interfaces and InternalLinks. Certainly, the data model lacks many ontology-specific entities like rules or constraints. However, the base models can be compared to each other and transformed. [4]

Fraunhofer IOSB's 'Production Monitoring and Control' business unit developed a prototypical engineering framework based on standardized communication and processing (OPC-UA) using a standardized data format (CAEX). [5] It allows for both the automated customization of the production monitoring and control system and the generation of the associated process visualization. The combination of both standards to form a framework underlines the strengths of each and opens up new potentials for the automation of automation.

The infrastructure of OPC-UA unifies all previous OPC-based technologies under a ‘platform-independent umbrella’. OPC-UA provides mechanisms for the standardized, asynchronous, distributed communication. [6], [7]

OPC-UA is a standard which accomplishes to support process communication in a structured way with an underlying user-defined information model. The information model enables users to create a representation of their plant using the object-oriented model paradigm. The OPC foundation therefore provides an XML schema for describing these models. Furthermore, the foundation defined a graphical representation for OPC-UA information models which is much more intuitive for users than the XML representation.
2 MOTIVATION
Currently, XML data described in CAEX has to be transformed manually to an OPC-UA-compliant notation. Thus, the current use of CAEX within the OPC-UA server is inefficient. The OPC-UA-address space data can be edited within an XML editor; but this bases neither on the special form of address space description, nor on the included semantic. For a better use of the standardized CAEX contents, the CAEX schema shall be integrated in the OPC-UA-server address space. In this way, the abilities of the server can be exploited. By means of an assistant tool, the CAEX data can be prepared for direct use within the OPC-UA components. This enables users to create a representation of their plant using the object-oriented model paradigm. The OPC foundation therefore provides an XML schema for describing these models. Furthermore, a graphical representation for these information models was defined which is more intuitive for users than the XML representation.

For the information model design, Fraunhofer IOSB developed a graphical editor. It allows to model graphically so called address spaces of OPC-UA servers which are the ‘living’ online representation of the underlying information model. The defined graphical model can be exported to an XML file which is compliant to the UA XML schema and can be imported into any OPC-UA server. But the UA Modeler goes even further, it can import CAEX information and transforms it to an OPC compliant description.

Thus, MES can take advantage of the skills of OPC-UA servers and CAEX data can be managed online. The developments shall be usable for different user groups and even within the World Wide Web. The mentioned concepts were tested by different use cases in combination with an application example exemplified hereinafter.

3 OPC-UA MODELER
3.1 Requirements
The developed tool shall support the following functionality: CAEX data shall be transformed into the graphical OPC-UA representation. OPC-UA-XML data shall be visualized graphically. The graphical representation shall be modeled according to the provided model by the OPC foundation. The base structure for graphical representation shall be present all the time. This is not necessary for the OPC-UA-XML data, but for the representation of the address space. The creation, positioning and change of graphical objects shall be supported. The graphical representation of the OPC-UA-address space shall be stored and exported in the OPC-UA-XML description.

The OPC-UA-address space modeler shall be evaluated by two use cases: First, the manual design and review of an address space and the export of the solution in OPC-UA-XML. Secondly, the import of a CAEX-XML file and its export to OPC-UA-XML, based on use case one. The defined use cases will be applied to an application example.

Therefore, a demo plant was modeled (see Figure 1). It consists of three types of plant components and to types of relations, the two conveyors
(blue rectangles) are called TB1 and TB2, the car (grey) is called Auto and the turntable (blue circle) is called DT1. The resources of this example are linked via a PPR-interface (product, process, and resource – dotted line named PPR) to the product, namely the car. To model topological links of one resource to another, a topological interface was used (dotted line, named Topologie). Topology will be represented by the set of corresponding links. Both conveyors are modeled as plants of the same type. The conveyors have only one example attribute: length. The turntable possesses two variables: diameter and length. The car has the variables length and width.

Figure 1: Application example

3.2 Architecture
By means of the application example, the example data is modeled. The CAEX description follows the CAEX-XML schema. At the same time the OPC-UA model was developed. This was also done by means of the application example. To get a graphical representation of the CAEX data, a transformation was defined. The graphical OPC-UA representation is used as the internal data model. In line with the development of the conceptual solution and for simplification, transformations were designed on type/class and schema level. Nevertheless, implementation took place on instance level (see Figure 2).

Figure 2: Overview concept development and implementation

3.3 Internal data model
The internal data is the same as the graphical representation proposed by the OPC foundation. Possible entities within this representation can be seen in Figure 3. By means of this representation, all possible elements of an OPC-UA address space and their connections between each other can be modeled. The model differs from a straight object-oriented modeling, because of the relations modeled as independent objects. This resembles
an ontology-based approach: a concept hierarchy, a property hierarchy and links between these elements.

Figure 3: OPC-UA notation for address space modeling [8]

Within each address space, there is a specific base structure, which arranges all created elements. There is for example always a root element where all other nodes are hung up (e.g. objects, object types, or relations). This base structure is not explicitly named within an OPC-UA-XML description of an address space, but it is included in each server and is depicted within the graphical modeling for reasons of completeness.

3.4 CAEX-Import
At Fraunhofer IOSB, the CAEX model is used with some extended schema-conform conventions and additional elements. One main point there is the division of all base components of the CAEX model in product, process and resource. This was also respected during the modeling of the application example in CAEX. The transformation of the CAEX model components to the graphical notation of the OPC-UA is done as shown in Figure 4.

This is not an 1:1-mapping. For example CAEX InternalElements represent plant instances. These are mapped to concrete objects within the OPC-UA address space. The CAEX SystemUnitClasses can be seen as types or classes and therefore correspond to OPC-UA ObjectTypes. The mapping doesn’t lead to a bijective transformation. Nevertheless, a CAEX export out of the OPC-UA modeler is not unambiguous.

3.5 OPC-UA-Import, -Modeling and –Export
The graphical representation is exported into OPC-UA-XML to generate an address space easily. The transformation for this can be seen in Figure 5. This is a 1:1-mapping. Each graphical object, whether if relation or entity, can create an XML fragment (an XMLNode) which describes the object itself. By means of a central method it is possible to unify all objects in a common XML description. This simplifies realizing an export of the
address space description. To import an OPC-UA address space appropriate objects and attributes are created from the XMLNodes.

<table>
<thead>
<tr>
<th>OPC-UA-XML</th>
<th>OPC-UA-XML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>Symmetric ReferenceType</td>
</tr>
<tr>
<td>ObjectType</td>
<td>Asymmetric ReferenceType</td>
</tr>
<tr>
<td>Variable</td>
<td>Hierarchical ReferenceType</td>
</tr>
<tr>
<td>VariableType</td>
<td>HasComponent</td>
</tr>
<tr>
<td>DataType</td>
<td>HasProperty</td>
</tr>
<tr>
<td>Object</td>
<td>HasSubtype</td>
</tr>
<tr>
<td>ReferenceType</td>
<td>HasTypeDefinition</td>
</tr>
<tr>
<td>Method</td>
<td>HasEventSource</td>
</tr>
</tbody>
</table>

Figure 5: Mapping of graphical OPC-UA model to OPC-UA-XML

### 3.6 Graphical User Interface

The interface between user and program is a very central point in this application. Thus, there were different requirements to be met by the application. On one hand, the graphical OPC model is used for modeling. On the other hand, operations for creation, change, deletion, and further handlings (drag&drop, etc.) are possible. According to the requirements in 2.1, the user interface shall include three object types: entities, relations, and the graphical workspace.

The logic of the implementation is based on XML, the Windows Presentation Foundation (WPF), and corresponding transformation logic. The graphical user interface (GUI) was developed by means of WPF, Silverlight 3 [8], Microsoft Expression 3 [9], and XAML.

Entities are divided into different OPC-UA types. An entity can be moved by drag&drop. If this entity is linked to another entity by a relation, its position shall change in line with mouse movement. A relation denotes a link of a specific type between two entities. So it can be called ‘reference’. A relation is moved by drag&drop and is visualized as a line where the type can be seen by means of different labels. The start, the center and the end of this line can be moved by drag&drop. If the start of a line is moved, the position of the start of the line changes, but not the position of its end. In the same way, the position of the beginning doesn’t change with a repositioning of the end. If the center of the line moves, the position of the complete line changes. If the start or the end of a line is dropped over an entity, a connection between entity and relation is created.

The workspace of the user interface includes instances of both other object types. In this way, the graphical representation can be created and modified. The deletion of entities is supported as well as their creation. If an entity is deleted, linked relations are deleted, too. Functions for positioning were developed as well. Therefore, additional menus (EntitySetting – A, RelationshipSetting – B, EntityMenu – D, RelationshipMenu – E, Figure 6) as well as the XML-Import- and -Export-
window are included. On top, there is an additional menu for the workspace (C, Figure 6). Within Figure 6
1. shows the controls for localization (switch German, English);
2. points to the controls for the creation of objects;
3. consists of functions to change the workspace, e.g. zoom or show/hide gridlines;
4. shows a legend of all possible entities and relation;
5. is the workspace itself.

Figure 6: Container class for workspace

4 SUMMARY AND OUTLOOK

In this contribution a graphical tool for modeling OPC-UA server address spaces is presented. The whole project took place in the field of production monitoring and control. For this field, the OPC-UA modeler helps to integrate CAEX into OPC-UA address spaces during the engineering. The evaluation of both use cases was successful. Figure 7 depicts an imported graphical description converted from CAEX.

This can be reviewed and edited now manually, and be exported into OPC-UA XML. By this way, the authors show that by means of the tool, integration of predefined semantics in form of CAEX data is possible. However, the results have to be checked against more complex examples and the development has to continue.

The OPC-UA address space modeler was developed for OPC-UA developers or users to apply more intensively the functionality of an OPC-UA server, to improve work with it, and to integrate CAEX.

Examples for new developments promoting and extending the modeler are enhanced positioning methods or online services such as the supply of online data.
5 REFERENCES


