# Lossless exchange of graph based structure information of production systems by AutomationML

Arndt Lüder, Nicole Schmidt, Sebastian Helgermann
Otto-v.-Guericke University
Universitaetsplatz 2
D-39106 Magdeburg
Germany
[arndt.lueder, nicole.schmidt]@ovgu.de

## Abstract

*Following the increasing complexity of production systems and increasing consideration of engineering costs engineering processes have to be efficient and should generate fault free results. As different engineering disciplines, engineering tools and humans are involved in the engineering process information exchange is a crucial factor calling for standardized and lossless technological solutions.*

*To avoid interpretation problems similar information should be exchanged by similar representations. As graph based structures very often can be seen within production systems the representation of graph based models within information exchange technologies should be unique.*

*Within this paper a first attempt towards the development of a unique modeling approach for graph based structures within the standardized data exchange format AutomationML will be considered.*

## 1. Motivation

Engineering of production systems is a time and skill intensive task where several engineering disciplines with their special knowledge and specialized tools are involved executing a complex engineering process [1, 2]. Beneath several structural and technological challenges of such projects resulting from the intended production system architecture and behavior [3] an important problem is the interoperability along the engineering process among engineering activities and engineering tools involved [4]. Especially the problem of information exchange among engineering tools is an important challenge as it strongly affects the efficiency and quality of an engineering process [5]. There are several approaches tackling this problem exploiting different technological solutions [6, 7, 8]. One of them is the AutomationML data exchange format [9].

The exploitation of standardized data exchange formats calls for the definition of unique mapping rules between the tool internal data structure and the data format ensuring an appropriate semantic transfer of the information [10].

There are two approaches known to ensure semantic clarity. The first one is the strategy to avoid any semantic interpretation range by defining for each data object the exact semantics. This is done for example in the STEP approach [7, 8]. Here a fixed amount of information can be transferred. If there is something to be transferred currently not defined it might be a problem. The second approach is the generic approach as used in AutomationML [9, 10]. Here each object will carry an indication of the semantic by its own. This approach enables the transmission also of new and prior unknown information sets. Nevertheless, the generic approach makes it possible to describe similar structures of production systems in very different ways. This may result in confusion of the users of the data exchange format. Thus, it seems to be promising to define reference structures for data modeling.

This paper intends to give a reference structure for another set of information, graph based structures. Graph based structures are very common in the engineering of production systems. Logistic networks, production plans, wiring networks, piping networks, communication networks, GOZINTO graphs, and much more are structured like graphs and can be modeled by graphs with annotations. Thus it seems to be reasonable to define a reference structure for graph representation by AutomationML enabling its application in the different engineering domains of the named information sets.

To reach this aim the paper is structured as follows. In section 2 graphs are defined more formally and an application case is described. In section 3 AutomationML is represented followed by the intended methodology for graph representation (section 4). With a summary the paper ends.

## 2. Graphs in production systems

A graph G = (V (G), E (G)) is defined by two non empty sets: vertex set V (G) and edge set E (G). These sets have the property that E (G) $\subseteq$ V (G) x V (G) holds, i.e. the vertices are linked by the edges (see Fig. 1) [11].

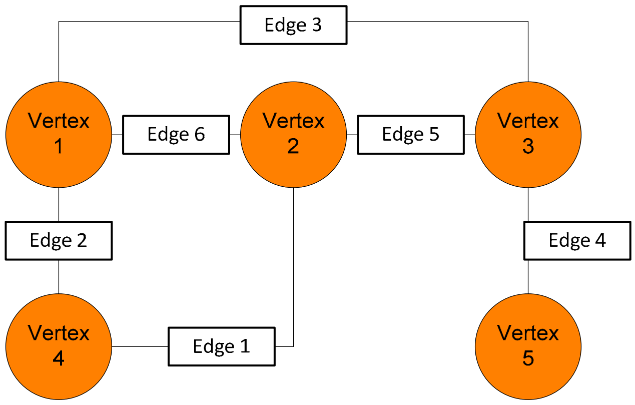


**Figure 1: Graph example**

If information is added to the objects of a graph it can be seen as labels of the related objects. Labels can have different forms, for example real numbers or even text boxes. For the development of graph models labels are one of the most important characteristics. For labeled graphs the definition above has to be extended.

A labeled graph LG = (V (G), E (G), $L_1$, $L_2$) is a graph with two additional mappings $L_1$, $L_2$. For the mappings holds that there are annotation sets $A_1$ and $A_2$ with $L_1$: V(G) $\rightarrow$ $A_1$ is a mapping into the vertex set on the annotation set 1 and $L_2$: E(G) $\rightarrow$ $A_2$ is a mapping into the edge set on the annotation set 2.

Graphs emerge in different application cases of production systems, e.g. in transportation systems.

A transport system consists of a route network, the cargo, the means of transport, stations (= sources and sinks) and the actual transportation process.

One possible way for modeling transport systems is the following. The vertex set of a transportation system graph model consist of vertices representing the stations of the transport. The edge set is divided into two sub-sets which typify the route network and the transportation process. The route network is formed by edges representing physical connections and the transportation process by edges expressing individual transport process. The identified properties of the elements are assigned to the elements of the graph (edges and vertices) in form of labels.

## 3. AutomationML

The AutomationML data exchange format is under development by the AutomationML e. V. [12] and currently within the international standardization process at IEC under the reference number IEC 62714. It is a neutral, open and XML based data exchange format and enables the consistent and lossless exchange of engineering information related to manufacturing system topology, geometry, kinematics, and control behavior and exploits an object oriented approach [9]. Each AutomationML object may integrate different information with different semantics related to different engineering disciplines. The AutomationML follows a modular structure and consists of different separated XML based data formats, which are combined under one roof so called top level format. Logically it is partitioned into descriptions of: plant topology (following CAEX 62424), geometry and kinematics (following COLLADA) and control related logic data (following PLCopen XML) were COLLADA and PLCopen files are referenced out of CAEX files.

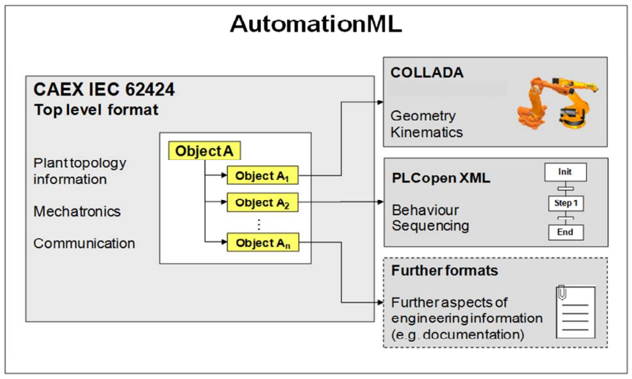


**Figure 2: AutomationML base structure [9]**

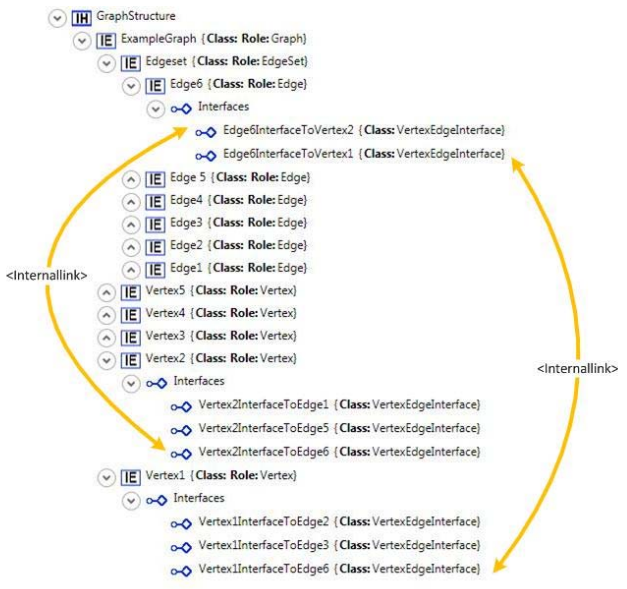
Theoretically, the same approach as for logic and geometry information can be exploited to model graphs in AutomationML. Existing graph representation formats like the Graph Exchange Language (GXL), the GraphML, and the eXtensible Graph Markup and Modeling Language (XGMML) can be exploited to be referenced. But this approach will strongly separate the graph model from the plant topology structure. This will duplicate modeling efforts. Thus, it is much more favorable to model graphs using the means for CEAX [13] and its defined application context in AutomationML.

## 4. Modeling of graphs using AutomationML

To model graph structures in AutomationML the most important class concepts of CAEX are exploited: role classes, interface classes and system unit classes [10].

### 4.1. Basics

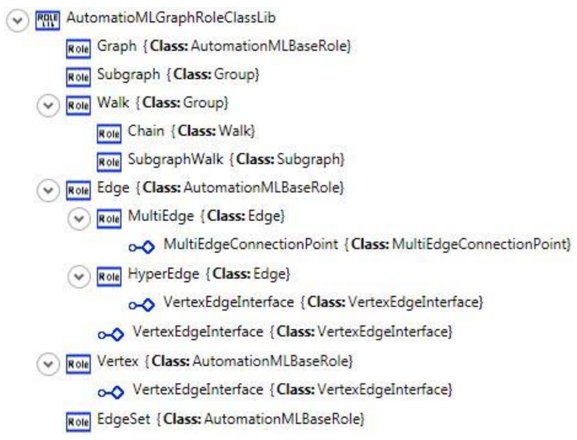
Starting point for the modeling approach is the definition of mapping rules to map the graph objects vertex and edge to AutomationML objects by use of CAEX. Thus, an <InternalElement> is generated as representative of the entire graph. Characteristics and additional information describing the graph, i.e. labels, can be attached to this element by means of attributes. Then the elements of the vertex set and edge set are created as child objects of the parent object graph. First of all, all vertices of the graph and their associated labels are transformed into the form of an <InternalElement> and its attributes. Afterwards edges are transformed in a similar way. For greater clarity it is suggested that the edge objects are created as child objects of additional <InternalElement> for the edge set. To express relations between vertices and edges interfaces are used. Therefore, all <InternalElement>'s representing vertices have as much interfaces as they have incident edges and all <InternalElement>'s representing edges usually have two interfaces. Interfaces of incident edges and vertices can then be linked by internal links. For the graph example from Figure 1 the resulting structure is given in Figure 3.



```
GraphStructure
  ExampleGraph {Class: Role: Graph}
    Edgeset {Class: Role: EdgeSet}
      Edge6 {Class: Role: Edge}
        Interfaces
          Edge6InterfaceToVertex2 {Class: VertexEdgeInterface}
          Edge6InterfaceToVertex1 {Class: VertexEdgeInterface}
      Edge 5 {Class: Role: Edge}
      Edge4 {Class: Role: Edge}
      Edge3 {Class: Role: Edge}
      Edge2 {Class: Role: Edge}
      Edge1 {Class: Role: Edge}
    Vertex5 {Class: Role: Vertex}
    Vertex4 {Class: Role: Vertex}
    Vertex3 {Class: Role: Vertex}
    Vertex2 {Class: Role: Vertex}
      Interfaces
        Vertex2InterfaceToEdge1 {Class: VertexEdgeInterface}
        Vertex2InterfaceToEdge5 {Class: VertexEdgeInterface}
        Vertex2InterfaceToEdge6 {Class: VertexEdgeInterface}
    Vertex1 {Class: Role: Vertex}
      Interfaces
        Vertex1InterfaceToEdge2 {Class: VertexEdgeInterface}
        Vertex1InterfaceToEdge3 {Class: VertexEdgeInterface}
        Vertex1InterfaceToEdge6 {Class: VertexEdgeInterface}
```

**Figure 3: Graph model to Figure 1**

To make the model semantically and structurally exact it is necessary to define appropriate role classes and interface classes.

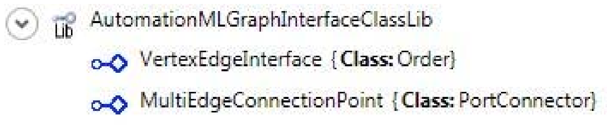


```
AutomatioMLGraphRoleClassLib
  Graph {Class: AutomationMLBaseRole}
  Subgraph {Class: Group}
  Walk {Class: Group}
    Chain {Class: Walk}
    SubgraphWalk {Class: Subgraph}
  Edge {Class: AutomationMLBaseRole}
    MultiEdge {Class: Edge}
      MultiEdgeConnectionPoint {Class: MultiEdgeConnectionPoint}
    HyperEdge {Class: Edge}
      VertexEdgeInterface {Class: VertexEdgeInterface}
    VertexEdgeInterface {Class: VertexEdgeInterface}
  Vertex {Class: AutomationMLBaseRole}
    VertexEdgeInterface {Class: VertexEdgeInterface}
  EdgeSet {Class: AutomationMLBaseRole}
```

**Figure 4: RoleClassLib for graph modeling**

The classes of the new <RoleClassLib> are representative of all objects that have been defined previously. As given in Figure 4 the defined roles cover all relevant objects like graph, vertex and edge. In addition roles can be defined for special purposes to represent special types of edges (like hyperedges) and special types of graphs (like chains).

In parallel, a new <InterfaceClassLib> is defined to contain the interface classes required to link edge objects with corresponding vertex objects. The <InterfaceClassLib> shown in Figure 5 contains basically two types of classes. The first class describes the connection between an edge and a vertex and the second class describes the additional interface of a multi-edge, called "MultiEdgeConnectionPoint".



```
AutomationMLGraphInterfaceClassLib
  VertexEdgeInterface {Class: Order}
  MultiEdgeConnectionPoint {Class: PortConnector}
```

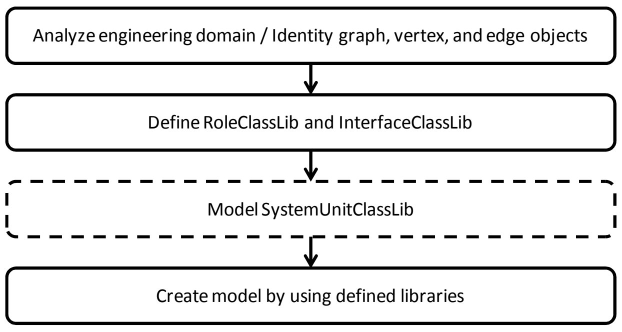
**Figure 5: InterfaceClassLib for graph modeling**

### 4.2. Application process

To use the previously described method for modeling graph based structures in application cases a four step procedure is required (see Figure 6).

In a first step the domain of interest is examined with respect to the planning objects usually used within this domain and its relations to each other. Based on this examination it has to be defined which objects will represent a complete graph, which objects are vertices and edges including their properties.

Next the appropriate role class library and the appropriate interface class library have to be defined. Here the roles for the objects have to be semantically fixed. For each relevant object an appropriate role is defined in the application case related role class library by deriving it from the roles graph, vertex or edge. For example for a piping system objects like pumps, valves, and vessels can result in roles derived from role class vertex and pipes can lead to a role pipe derived from edge. In addition the necessary interfaces to link edge and vertex objects have to be defined analogously.



Analyze engineering domain / Identity graph, vertex, and edge objects
Define RoleClassLib and InterfaceClassLib
Model SystemUnitClassLib
Create model by using defined libraries

**Figure 6: Procedure for use of the graph modeling method**

In step 3 (which is optional) meaningful SystemUnitClasses for the engineering domain can be identified and modeled as templates for further use.

Finally, the defined classes can be used to model a practical system.

## 5. Summary and outlook

Graph based structures are wide spread in practical application cases. Most prominent examples are piping systems, wiring systems, communication systems, or transportation networks. But there are several additional cases leading to graph based structures like the GOZINTO graph of product design, the business contact structure of a company set, or the work schedule of employees.

Within this paper a method has been proposed useable to model graph based structures in a syntactically and semantically unique way exploiting the data exchange format AutomationML. It enables the lossless and clear interpretable exchange of engineering information representing graph based structures.

In [14] several application domains are considered on a conceptual level and it has been shown that the proposed approach is generally applicable.

In the second approach the method has been practically applied by a working group of the AutomationML e. V. to develop the representation of communication systems based on AutomationML [15].

In future AutomationML e. V. will exploit the defined methodology to different application cases. Most relevant candidates are electrical wiring and enclosure design.

**References**

[1] A. Fay: Effizientes Engineering komplexer Automatisierungssysteme, Within E. Schnieder, T. Ständer (Editors): 20 Jahre - vom IfRA zum iVA - Jubiläumskolloquium, ISBN 978-3-9803363-0-7, 2009, S. 43–60.

[2] A. Lüder, M. Foehr, L. Hundt, M. Hoffmann, Y. Langer, St. Frank: Aggregation of engineering processes regarding the mechatronic approach, 16th IEEE Intern. Conf. on Emerging Technologies and Factory Automation (ETFA 2011), Toulouse, France, September 2011, Proceedings-CD.

[3] D. Wünsch, A. Lüder, M. Heinze: Flexibility and Re-configurability in Manufacturing by Means of Distributed Automation Systems – an Overview, in H. Kühnle (Editor): Distributed Manufacturing, Springer, 2010, pp. 51-70.

[4] Ch. Diedrich, A. Lüder, L. Hundt: Bedeutung der Interoperabilität bei Entwurf und Nutzung von automatisierten Produktionssystemen, at – Automatisierungstechnik 59 (2011), Issue 7, pp. 426 – 438.

[5] L. Hundt, A. Lüder, E. Estévez Estévez: Engineering of manufacturing systems within engineering networks, 15th IEEE Conf. on Emerging Technologies and Factory Automation (ETFA 2010), Bilbao, Spain, Sep. 2010, Proceedings-CD.

[6] T. Moser, R. Mordinyi, D. Winkler, M. Melik-Merkumians, S. Biffl: Efficient automation systems engineering process support based on semantic integration of engineering knowledge, 16th IEEE Conf. on Emerging Technologies & Factory Automation (ETFA), Toulouse, France, Sep. 2011, Proceedings.

[7] X. Xu, A. Yeh C. Nee: Advanced Design and Manufacturing Based on Step, Springer, 2009.

[8] Fiatech: An Introduction to ISO 15926, November 2011, http://www.fiatech.org/images/stories/techprojects/project_deliverables/iso-intro-ver1.pdf

[9] R. Drath (Editor): Datenaustausch in der Anlagenplanung mit AutomationML, Springer, 2010.

[10] R. Drath, M. Barth: Concept for managing multiple semantics with AutomationML - maturity level concept of semantic standardization, 17th IEEE Conf. on Emerging Technologies & Factory Automation (ETFA), Krakow, Poland, Sep. 2012, Proceedings.

[11] R. Balakrishnan, K. Ranganathan: A Textbook of Graph Theory, Springer, 2012.

[12] AutomationML: Website, www.automationML.org, last access March 2013.

[13] Industrial Elecronic Commission: IEC 62424:2008, Representation of process control engineering - Requests in P&I diagrams and data exchange between P&ID tools and PCE-CAE tools

[14] S. Helgermann: Entwicklung einer generischen Darstellungsmethode für grafenbasierte Strukturen von Produktions- und Automatisierungssystemen mittels AutomationML, Bachelor thesis, Otto-von-Guericke University Magdeburg, Fakulty for Mechanical Engineering, October 2012.

[15] A. Lüder, M. Riedl, R. Drath, O. Niggemann, B. Heines: Austausch von Entwurfsdaten für Kommunikationssysteme mit Hilfe von AutomationML, Automation 2013, Baden-Baden, Germany, June 2013, Proceedings, VDI Verlag, VDI-Berichte.