Implementation of an AutomationML-Interface in the digital factory simulation

M.Sc. Ender Yemenicioğlu
tarakos GmbH
Werner-Heisenberg-Str. 1
D-39106 Magdeburg
yemenicioglu@tarakos.com

Prof. Dr. –Ing. habil. Arndt Lüder
Otto-von-Guericke University
Universitätsplatz 2
D-39106 Magdeburg
arndt.lueder@ovgu.de

Abstract

The role of virtual commissioning and manufacturing plant simulation in the process planing is getting more important every day (Reinhart & Wünsch, 2007). However the heterogenous environment of engineering tools and the need of data exchange between them cause big costs and time waste. (Kiefer, 2008) A concept to transfer information types like topology, geometry, kinematics, logic, physical behavior is currently one of the main requirements of the mechatronical engineering process. In this paper, a practical attempt of solving the data exchange problem with the help of AutomationML® format will be introduced.

Motivation

The term „digital factory“ is defined as a network of digital models and methods which among others consists of simulation and 3D visualization. Its purpose is the integrated planning, implementation, control and continuous improvement of all essential plant processes and resources in connection with the product. (VDI, 2008)

Many engineering fields, separate software tools and their associated data models are used in the engineering chain of production planning, which contains disciplines from technical, economical and natural sciences. It is required to have project wide observation and controlling independent of individual engineering tools. Therefore data management is one of the major tasks within digital factory. (Lüder, Rosendahl, & Schmidt, 2013)

Essential basement of the digital factory are digital data models of all relevant components of a production system including products and production systems covering structure, behavior, and relations/dependencies. Thus, one required input is a visual scene of a production system modeling the structure, geometry, kinematics and behavior of the production system in a way applicable for simulation based analysis. To create such a scene is usually a task of the OEMs or the production system developer / user.

The accumulated data can further be used by other software tools, which are specialized in a certain field like mechatronic, physic simulations, control design, virtual commissioning or maintenance. Therefore, data exchange is an important part of production system engineering exploiting the digital factory. Nevertheless this data exchange is also a cost factor in the workflow (Drath, Lüder, Peschke, & Hundt, 2008) as bad implementations of this data exchange (for example exploiting only PDF files or proprietary data formats) can cause errors and redundant engineering actions.

AutomationML, which combines the topology, geometry, kinematic, logic information as well as possibly other XML based information, has the potential to overcome the problems by providing a generally accepted data exchange format able to model and transfer the required data.

However, the format itself does not ensure the quality of exchanged information. Errors such as unique identity key violation or defective normal vectors in geometry are not uncommon. The evaluation of external references is also a challenge for most importing tools. These aspects should be

taken into consideration during the development of an interface. This work presents a practice of implementing AutomationML-format importing and exporting functionality for the purposes of digital factory simulation in taraVRBuilder and taraVRControl software tools.

An essential part of the digital factory is the virtual commissioning. It is related to the validation of production system behavior before system physical implementation to avoid system malfunctioning (Reinhart & Wünsch, 2007). Virtual commissioning requires information from system engineering, mechanical engineering, electrical engineering, and control engineering (at least) and is therefore the ultimate example for required data exchange within the engineering of production systems.

Definition of task

The particular steps of data exchange on the way to virtual commissioning should be clarified to define the task considered in this paper. Mostly the need of a new product is first determined, and process engineers make a concept design for the manufacturing tools and workflow. If the geometry and kinematic information of manufacturing tools already exist, it can be transferred to layout planning tools. If not, it should be created with the help of 3D engineering tools. The layout plan can also be transmitted as manual sketches or 2D drawings to a layout planning tool. With the help of these information a 3D scene of the manufacturing plant can be created.

The outcome of the layout planning should be transferred to other engineering tools, which are specialized in particular fields: PLC programming tools, robot programming tools, electrical design tools, physical behavior definition tools, etc. Theoretically all of these tasks can be fulfilled parallel, and the outcome of each process can be merged into one container file.

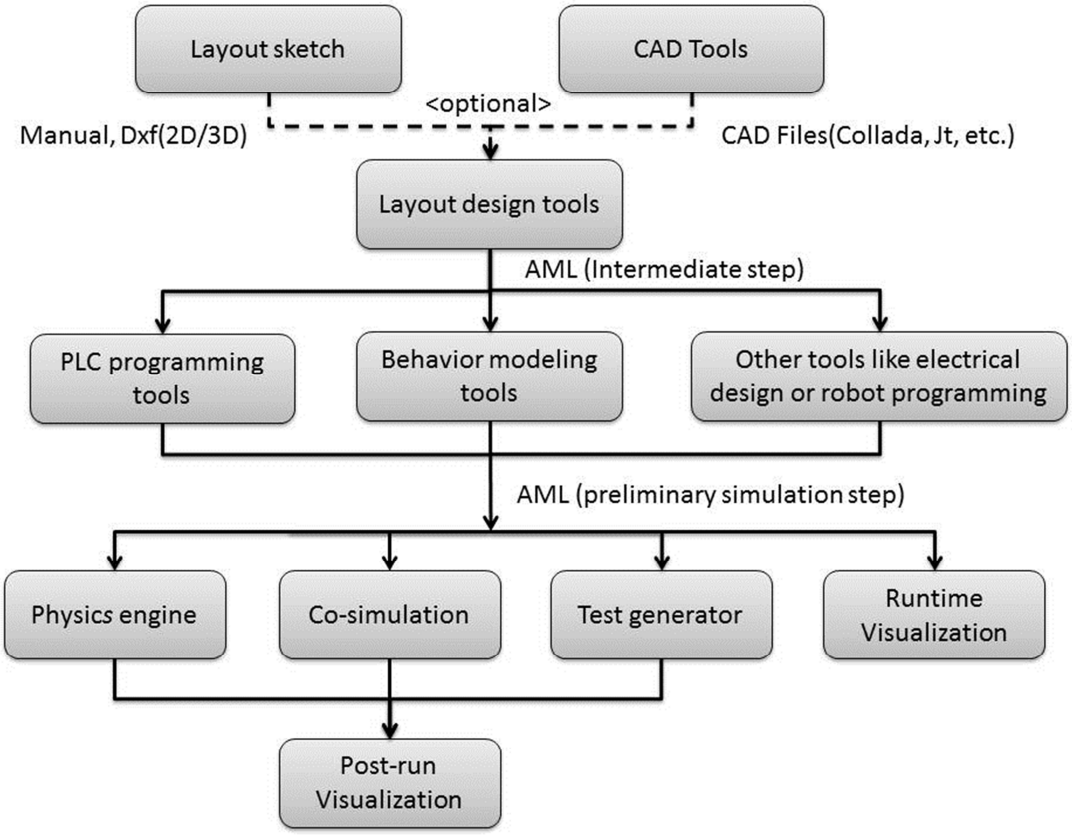

Figure 1 Plant simulation process and implementation of AutomationML®

The result data can be used for the purpose of creating a virtual scene with physics engine for a realistic visualization. With the help of co-simulation other engineering solvers can be added to the simulation. An automatic test generation is also a possible task, which can be supplied with the result of

co-simulation. Simulation results can be visualized at real-time (runtime visualization) or a post run visualization could be necessary because of performance issues and to provide high fidelity of the outcome. Post run visualization is more realistic than the runtime visualization because the simulation output is more accurate and precise due to longer timing constraints.

But the problem is that there is no existing usage of a data exchange format, which supports all these steps in the simulation planning process. As a neutral data exchange format which includes the different facets of plant engineering AutomationML® is a strong candidate to be the solution. However an implementation of import and export functions for the above explained tool chain is needed. (cf. Figure 1)

In the following the example of a tool chain used for behavior simulation within virtual commissioning is considered.

Components of exchanged information

In a simulation process a manufacturing plant consists of several components, which includes mechanical, electrical, automation relevant and other information aspects. These components can be named as mechatronical engineering objects. Mechatronical engineering object is the representation of a mechatronical unit in the real component, which is “the combination of energy, information and material flows that are connected via sensor and actuators to an information processing unit, usually establishing a hierarchical structure.” (Foehr, Lüder, Hundt, & Wagner, 2010)

A conveyor unit from materials handling field can be taken as a simple example of a mechatronical engineering object. The main geometrical properties of a conveyor are the length, width, height and degree of bending angle. The kinematic property is the direction of the material flow. A conveyor consists of load-carrier system (e.g. Rollers) and a frame with supporting legs. They can be considered as sub-components. The connection points with other system components should also be defined for a conveyor object. (cf. Figure 2)

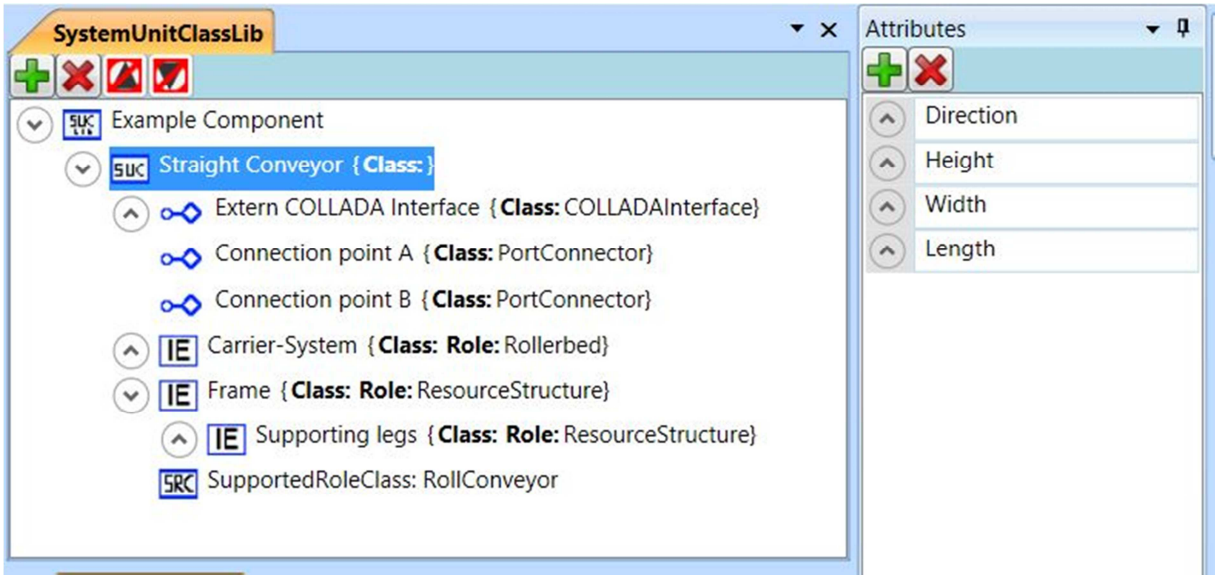

Figure 2 View of an example conveyor object in AutomationML®-Editor

As easily visible in this example, the relevant information within the considered application of virtual commissioning are geometry and kinematics information of the system components and its geographical relations describing the visual scene and the behavior information modeling the controlled and the uncontrolled behavior of the production system (Lüder, Rosendahl, & Schmidt, 2013). These information sets have to be exchanged.

Functional requirements

There are two main functional requirements in the tool chain: 1) Describing and exporting the own data model as AML-file, 2) Importing an AML-file from different type of sources.

For a successful export function, the quality of the exported information should be considered as an important factor. Not only the position and the geometry of the object but also the hierarchical structure and workflow should be transferred. For the identification of objects all exported components must have a unique identifier and they are not allowed to be changed in the further phases of tool chain. (Drath & Schleipen, Grundarchitektur: das Objektmodell, 2010) The tool itself should have an inner mapping of the objects and their identifiers for the recognition of the own objects after the export. The versioning of the file should also be supplied.

Inside the tool chain three types of importing tasks can be defined: 1) Importing the own created files (i.e. from taraVRBuilder to taraVRControl), 2) Importing a file which is created by own tool but changed from another tool afterwards (i.e. exporting from taraVRBuilder, further editing by an external tool and importing to taraVRControl), 3) Importing a file from a completely foreign origin.

It can be guessed that importing an own file is the easiest task among these. The scene can be created with the known hierarchical structure and the objects can be established via intern mapping of identifications. The possible changes in the attributes could be applied with known intern software methods.

The second type of importing is more challenging than the first one. It is possible that some unknown attributes or properties are additionally in the scene, and they should not be overwritten during the further work in the tool.

The third type of import is the most difficult one. But it does not mean there are no information in the file which can be recognized. The topology of the internal elements can be definitely identified. The position of the objects can be determined with the help of element attributes, which are derived from standard role class “Frame”. “Frame” role class defines the relative positions and the rotations compared to the parent object’s coordinate system. The geometry of the object is defined with a standard external reference interface “COLLADAInterface”. The logic behavior definition can also be recognized with the standard interface “PLCopen-XMLInterface” Further properties and structures, which are defined in the whitepaper (for example connections of objects via port interface and so on), could also be found out. (AutomationML consortium, 2013.)

Co-simulation requirements

The digital factory simulation requires the modeling and solving of behavior for multi-disciplinary systems, which cannot be achieved with only one simulation tool. For complex physical behaviors there are some specific modelling languages like object-oriented Modelica or data flow based graphical language Simulink. (Moriz, et al., 2011)The software tools which can solve these models and equations should work together with other simulation components like visualization, physics-engine, FEM-Solvers, etc.

Co-simulation is an approach for coupling the simulation systems where each specific tool engages with one part of the task. Intermediate outcomes such as variables and status information are exchanged between these tools after a predefined time step. A co-simulation master is responsible for coordinating the transfer of the data. The sub-systems are defined as slaves in master-slave concept.

The Functional Mock-up Interface (FMI) is an interface standard for such a co-simulation. Functional Mock-up Unit (FMU) is a component implementing the FMI. It consists of a zip package which con-

tains an XML description file and the implementation in source code as C language or as binary form. (Bastian, Clauß, Wolf, & Schneider, 2011)

For cooperation with physical solvers a FMU object can be referenced in AutomationML® as an external reference like for COLLADA and PLCOpen-XML. A new interface class "FMUInterface" which is inherited from the standard "ExternalDataConnector" can be implemented for this purpose. For further explanation see (Moriz, et al., 2011) and (Graeser, Kumar, Niggemann, Moriz, & Maier, 2013)

Implementation of the data exchange functionality

In this paper the plant topology and the geometry of the objects are the main interest points for the implementation, as the practical work is first succeeded for the visualization of these aspects for taraVRControl and taraVRBuilder tools. The future implementation of visualization with physics-engine interface and co-simulation environment will not be discussed further in this chapter.

For the realization of AutomationML® data exchange functionality a total of seven C#.NET libraries are used. First one is a foreign library named AutomationML®-Engine, which is provided by AutomationML® consortium for free use under the terms of the GNU Library General Public License. One library is used to convert the inner structure of the own software into CAEX instance hierarchy, and another one is to gather the inner structure back from the CAEX file. The other 4 libraries are used to convert the geometry between COLLADA and native geometry format (in this case VRML-Virtual Reality Modeling Language). The interactions between the libraries can be seen in Figure 3.

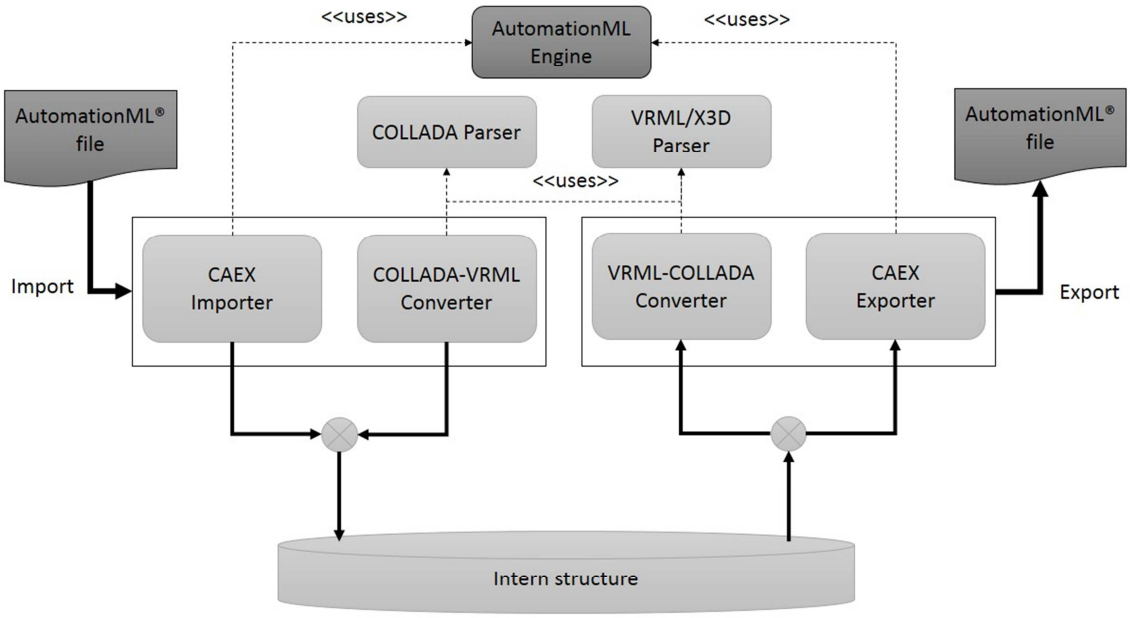

Figure 3 Software components of the AutomationML® importing and exporting functionalities

The import function parses the CAEX file with the help of AutomationML-Engine, goes through the "Frame" attributes to position the objects, if an extern COLLADA file is referenced, converts the geometry to the VRML format with the help of own converter functionality (see Figure 4).

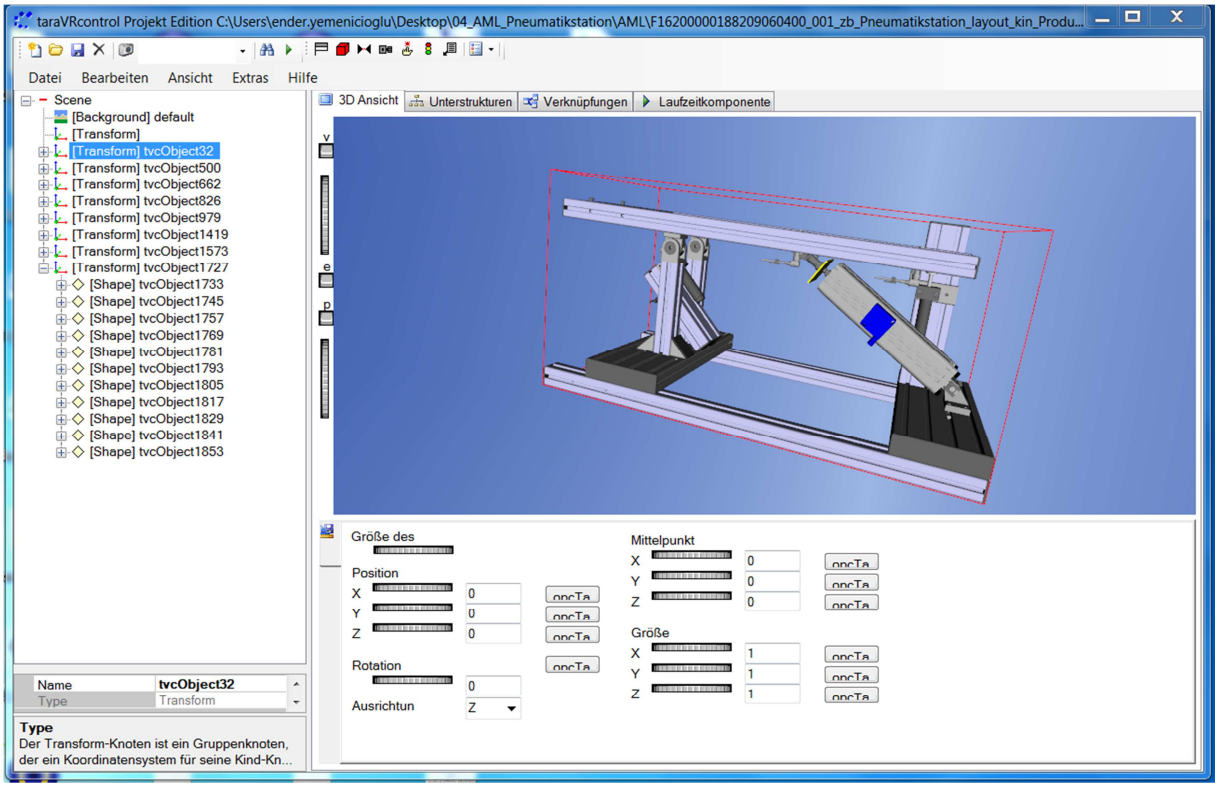

Figure 4 Imported pneumatic station demonstrator from AVANTI-Project in taraVRcontrol (AVANTI, 2014)

By export function the quality of the outcome is relatively higher; the hierarchy of objects is represented as instance hierarchy in CAEX format. A system unit class library, which contains the collection of own object library components, is used for the classification of instance objects, which is referenced as an extern file in the instance hierarchy file (see Figure 5). The role classes are also referenced externally. The connection points are defined as CAEX “InternalLink” with the help of interface class “PortConnector”. The position of the objects are written as “Frame” attribute in the instance hierarchy, and the geometry is converted to COLLADA via own software components and attached as extern reference in the file.

```
<ExternalReference Path="\Libs\UnitClassLibraries\tarakosSystemUnitClassLib.aml" Alias="tarakosSystemUnitClassLib" />
<ExternalReference Path="\Libs\InterfaceLibraries\AutomationMLInterfaceClassLib.aml" Alias="AutomationMLInterfaceClassLib" />
<ExternalReference Path="\Libs\RoleClass Libraries\AutomationMLBaseRoleClassLib.aml" Alias="AutomationMLBaseRoleClassLib" />
<InstanceHierarchy Name="Test">
    <InternalElement Name="Test_Id" ID="521fc77c-6a20-4196-adca-0d8c633af36b">
        <InternalElement Name="Gerade multistau" ID="0aab8254-0e31-4969-beed-e30e909a50d0" RefBaseSystemUnitPath="tarakosSystemUnitClassLib/Gerade_MultiStau">
            <Attribute Name="Breite">
                <Value>10000</Value>
            </Attribute>
            <Attribute Name="Laenge">
                <Value>5000</Value>
            </Attribute>
            <Attribute Name="Frame">
                <Attribute Name="x">
                    <Value>1</Value>
                </Attribute>
                <Attribute Name="y">
                    <Value>2</Value>
                </Attribute>
                <Attribute Name="z">
                    <Value>3</Value>
```

Figure 5 Extern reference for SystemUnitClassLibrary file

A challenging task in the implementation is the geometry conversion between COLLADA and native format VRML. COLLADA primitive geometry types, which are defined in “mesh” element, can be transferred directly as indexed geometry types under “shape” element in VRML. However the indices of the coordinates, normal vectors, texture coordinates and color vectors should be calculated again because one to one convergence is not possible. Triangulation methods are used for VRML geometry types like “box”, “cone”, “cylinder” and “sphere” to represent them as triangle/polygon meshes in

COLLADA. Effect properties like color, ambiance and transparency are parallel in “appearance” in VRML and “library_materials”, “library_effects” in COLLADA. Even so the differences in the representation should be regarded.

Discussion and Conclusion

Within this paper an implementation of data exchange interface with the purpose of integrating a plant layout planning tool to engineering chain of digital factory simulation is presented. The solution is based on the standardized data exchange format AutomationML. Description of hierarchical object structure of the plant layout and converting the geometry information were the subjects of main effort till now.

During this practice some bottlenecks are noticed too. The boundaries of the conversion between COLLADA and VRML/X3D is reached quickly if boundary representation (BREP) should be handled, which is supported in COLLADA but not yet in X3D. Non-uniform rational basis spline (NURBS) are existent in both formats, however it is rare to see in industrial applications. In case of need a comparison of syntaxes in both formats for NURBS should be examined. The kinematic properties of the elements in COLLADA are also a challenge for the transfer. “Rigid body components” in X3D specification offers a solution for the kinematic information, but the tool support for these components are not yet sufficient.

Another requirement is to enhance the data exchange of plant topology in the tool chain. The classification and transformation of objects in AutomationML® are carried out with the help of role class libraries. The identification of an imported object or the compatibility of an exported object to external tools is only possible if the role class of the object is defined and the role is known for the related tools. (Hundt & Prinz, AutomationML - Datenaustausch, 2013) Therefore there is a need of developing a common role class library for the specific purposes such as material flow, logistics, etc. which are not yet wide defined in the whitepaper of AutomationML®. However these task cannot be fulfilled without the participating of all partners in the process tool chain.

The data exchange occurs till now only unidirectional, that means a new scene is created with the import function and with export the previous status of the data is lost. All objects have a unique signature as “Globally Unique Identifier” (GUID) and header information for the project identification is also provided. The future steps to allow bidirectional data exchange is a support of versioning in the whole file and each instance objects. A “ChangeMode” attribute for the objects, which shows the status of the object if it is “created”, “changed” or logically “deleted”, can be implemented to track the changes in object-level. In this consideration no object is allowed to be erased physically from the file and the changed objects should exist with its former and actual form. (Hundt & Prinz, AutomationML − Engineering Workflow, 2013)

The next step for the defined task is creating a physical scene with help of accumulated data. An adapter, which describes the elements defined in AutomationML® as objects for the data model of physics engine, is in progress. An interface to communicate with the co-simulation environment is also required, which means a content pipeline to FMI should also be implemented.

References

AutomationML consortium. (2013.). *AutomationML Whitepaper. Part 1 - Architecture and general requirements.* Retrieved from www.automationml.org.

AVANTI. (2014). *Test methodology for virtual commissioning based on behaviour simulation of production systems.* Retrieved from http://avanti-project.de

Bastian, J., Clauß, C., Wolf, S., & Schneider, P. (2011). *Master for Co-Simulation Using FMI.* 8th International Modelica Conference, Dresden.

Drath, R., & Schleipen, M. (2010). Grundarchitektur: das Objektmodell. In R. Drath, *Datenaustausch in der Anlagenplanung mit AutomationML* (p. 55). Springer-Verlag Berlin Heidelberg.

Drath, R., Lüder, A., Peschke, J., & Hundt, L. (2008). AutomationML – the glue for seamless Automation Engineering. *Emerging Technologies and Factory Automation, 2008. ETFA 2008. IEEE International Conference*, (pp. 616-623).

Foehr, M., Lüder, A., Hundt, L., & Wagner, T. (2010). *Manufacturing System Engineering with Mechatronical Objects.* Otto-von-Guericke University.

Graeser, O., Kumar, B., Niggemann, O., Moriz, N., & Maier, A. (2013). AutomationML as a Shared Model for Offline-and Realtime-Simulation of Production Plants and for Anomaly Detection. In *Informatics in Control, Automation and Robotics* (pp. 195-209). Springer Berlin Heidelberg.

Hundt, L., & Prinz, J. (2013, March 19). AutomationML - Datenaustausch. *SPS Magazin 4*. Retrieved from http://www.sps-magazin.de/?inc=artikel/article_show&nr=72991

Hundt, L., & Prinz, J. (2013, October 29). AutomationML − Engineering Workflow. *SPS Magazin 11*. Retrieved from http://www.sps-magazin.de/?inc=artikel/article_show&nr=80349

Kiefer, J. (2008). *Mechatronikorientierte Planung automatisierter Fertigungszellen im Bereich Karosserierohbau.*

Lüder, A., Rosendahl, R., & Schmidt, N. (2013). Validation of behavior specifications of production systems within different phases of the engineering process. *Emerging Technologies & Factory Automation (ETFA).* IEEE 18th Conference.

Moriz, N., Faltinski, S., Graeser, O., Niggemann, O., Barth, M., & Fay, A. (2011). Integration und Anwendung von objektorientierten Simulationsmodellen in AutomationML. In *Automation 2011* (pp. 37-40).

Reinhart, G., & Wünsch, G. (2007). Economic application of virtual commissioning to mechatronic production systems. In *Production Engineering 1.4* (pp. 371-379).

VDI. (2008). *Digitale Fabrik Grundlagen VDI-Richtlinie 4499, Blatt 1.* VDI-RICHTLINIEN. Retrieved from www.vdi.de