



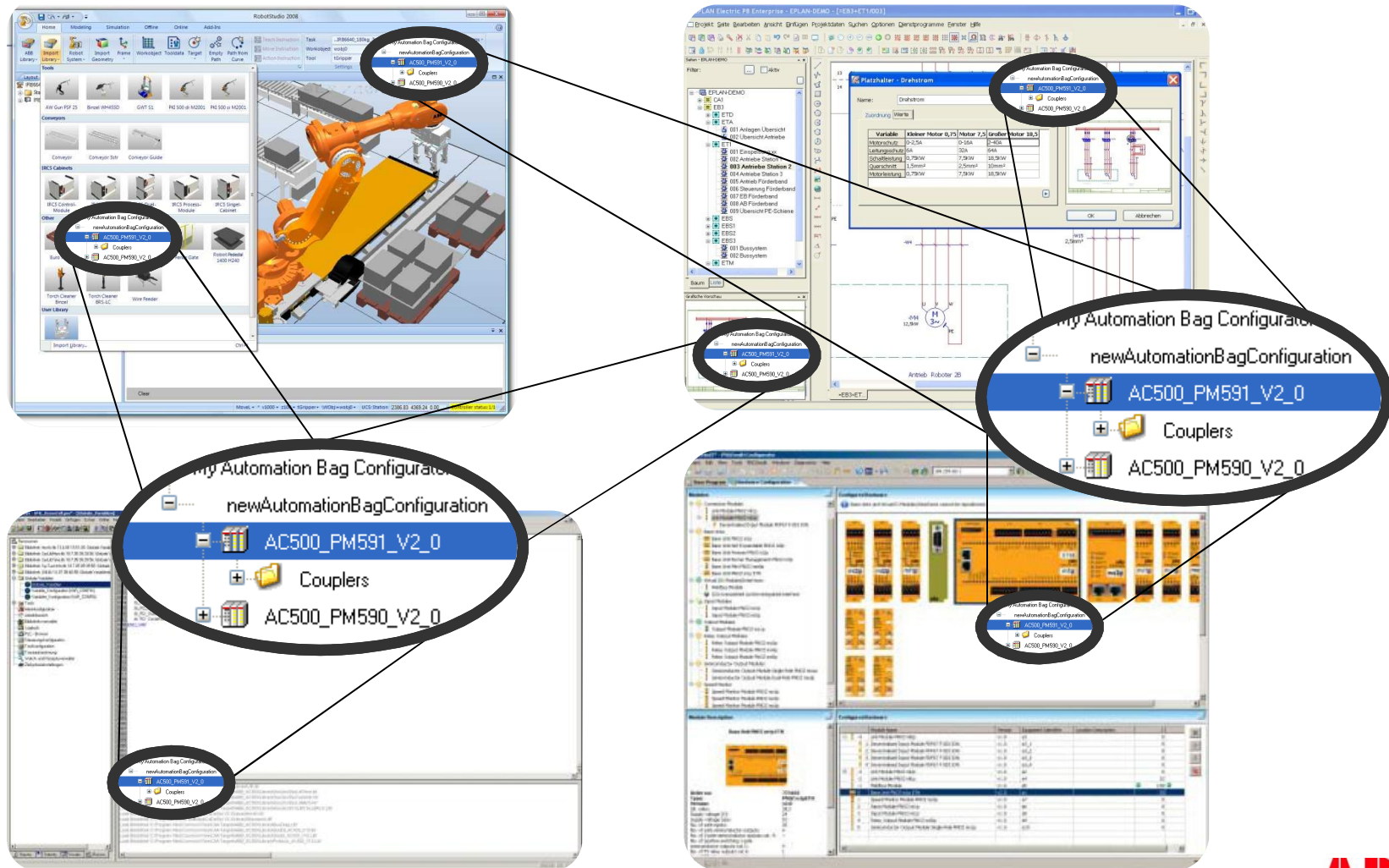
Dr.-Ing. Rainer Drath, ABB Corporate Research, AutomationML Plugfest 14.+15.10.2015

Lets Talk AutomationML

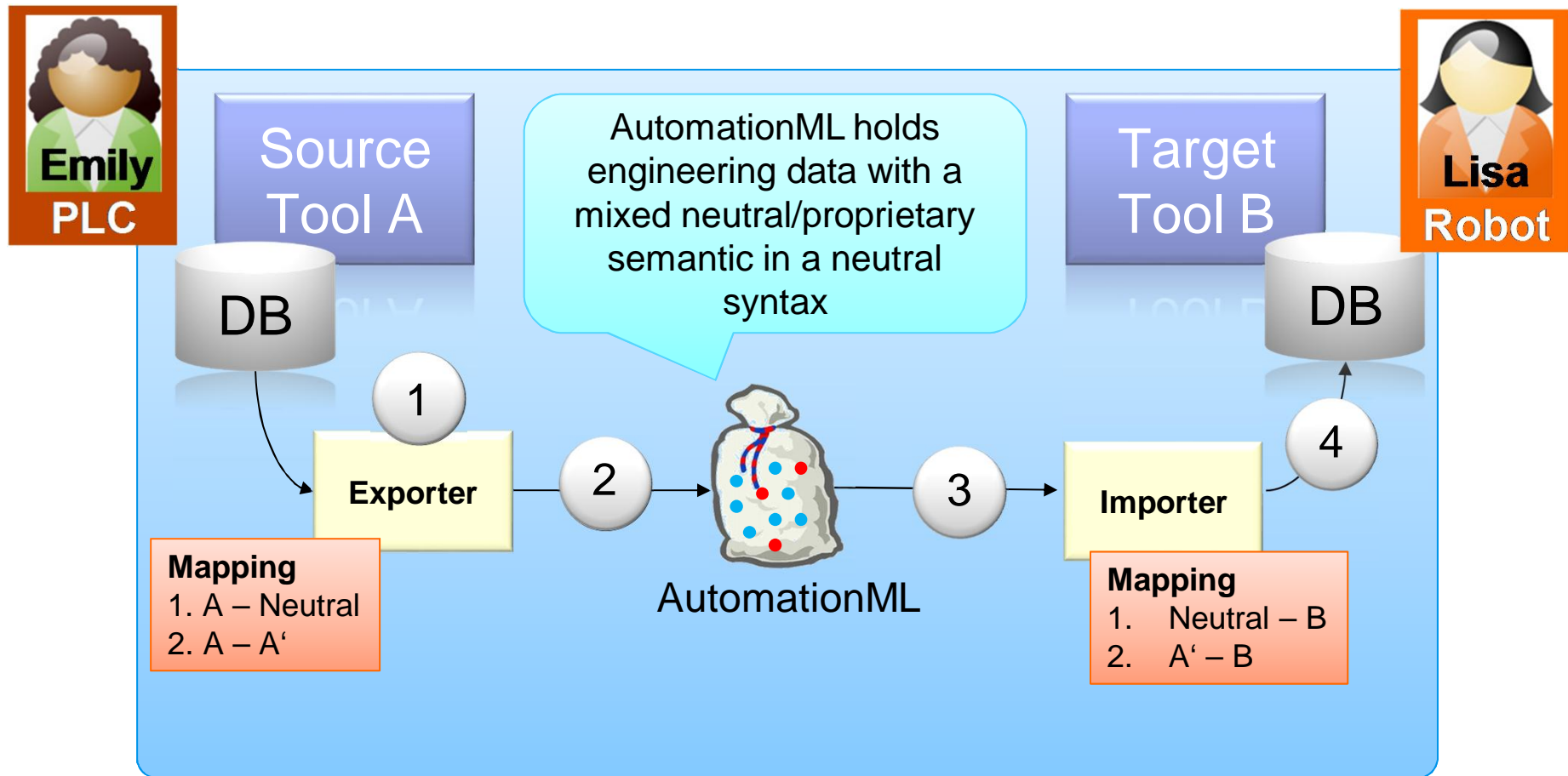
What is the effort of AutomationML programming?

Drivers behind AutomationML

Data exchange in a heterogeneous tool landscape



Data exchange scenario



The transfer of research into industry highly depends on applicability

Don't forget usability and transparency.
The value of knowledge scales with its application. An easy to use
software may generate more re-use than the corresponding PhD book.

This speech is about minimizing the AutomationML development effort

The scientific value of this research is finding a suited bridge between
AutomationML ideas and their application in industry and academia

Ways to program AutomationML:CAEX

The stony way

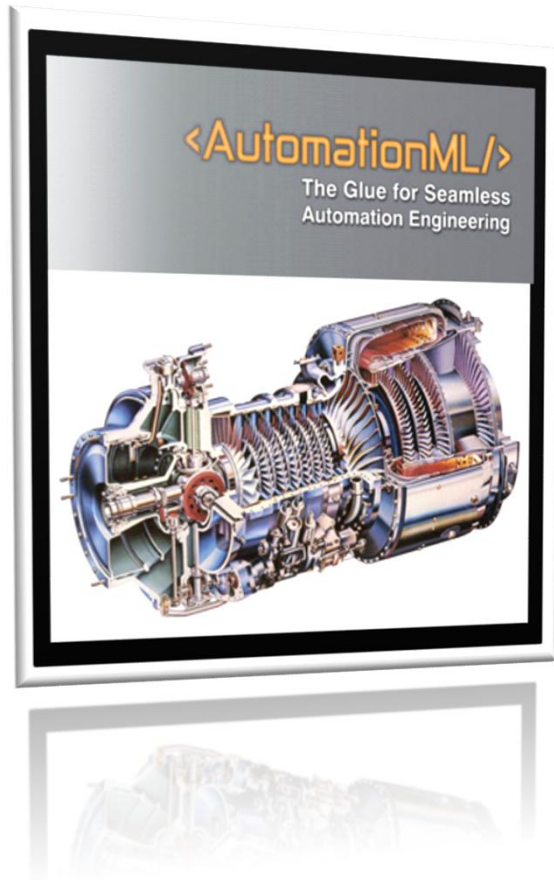


CAEX
IEC 62424

- Learn the CAEX data format according to IEC62424
- Learn the AutomationML standard according to IEC62714 (CDV)
- Use a XML parser (DOM) to read, write and change CAEX objects
- Always ensure that your software fulfills all standard requirements

Ways to program AutomationML

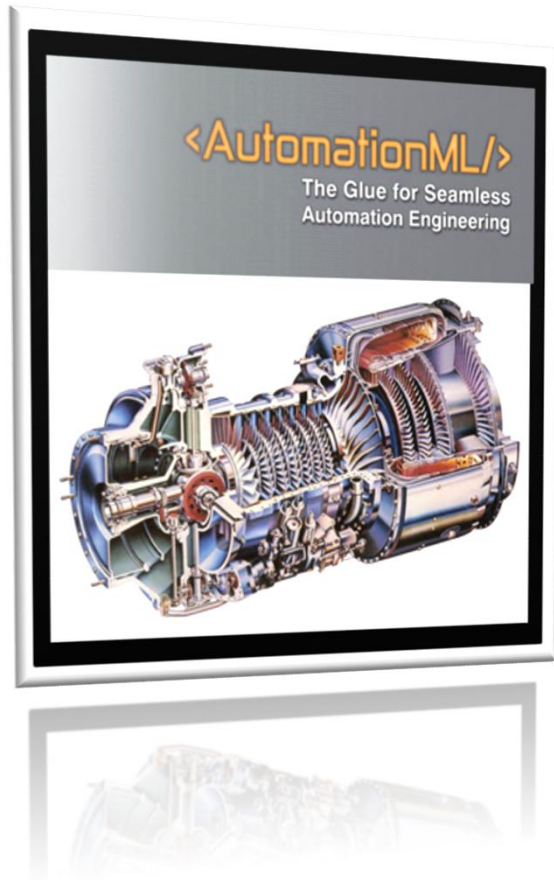
The easy way: use the AutomationML-Engine



- Exactly mirrors the CAEX data model into a C# class model
- Mostly automatically created
- Provides software functionality to **create**, **manipulate** and **validate** CAEX files according to IEC62424 and IEC 62714.
- Is the software foundation for a variety of AutomationML software, e.g. for the AutomationML Editor, importer/exporter in the AutomationML community

CAEX programming

Use cases



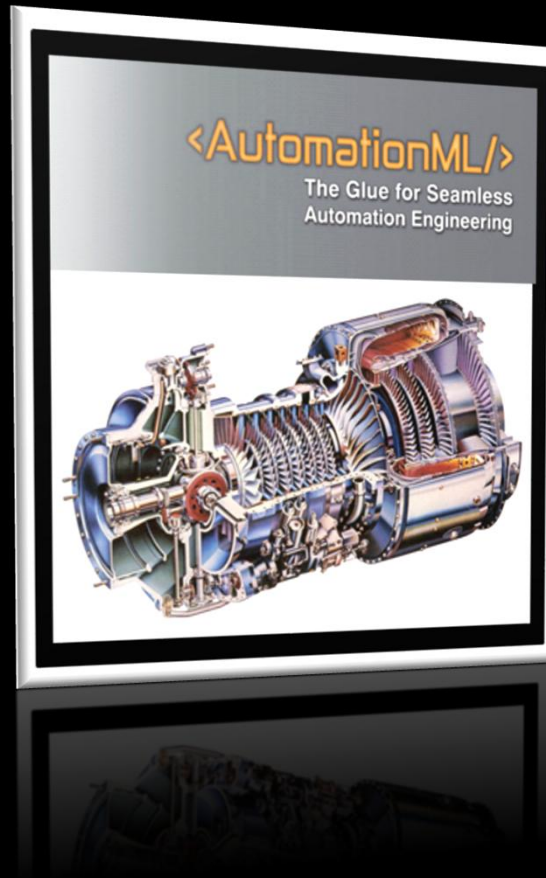
Basic use cases

- Creating a new CAEX document
- Opening an existing CAEX document
- Saving a CAEX document
- Creation of hierarchies, objects and attributes

Extended use cases

- Basic design of an Exporter
- Basic design of an Importer
- Error checking and automatic error correction

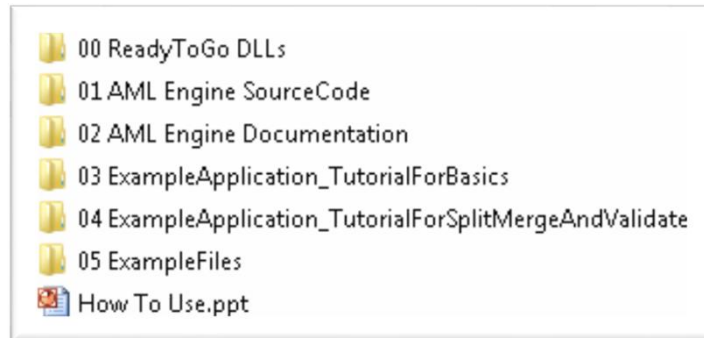
AutomationML programming



Basic use cases

AutomationML Engine

Folder Structure of the AutomationML Engine zip file



00 ReadyToGo DLLs

contains ready to go dlls for direct binding into your .Net projects

01 AML Engine Source Code

for AutomationML e.V. members only

02 AML Engine Documentation

contains a chm help file

03 ExampleApplication_TutorialForBasics

contains an example application demonstrating

- how to create a CAEX File
- how to save and store a CAEX file
- how to validate a CAEX file
- how to label a CAEX file

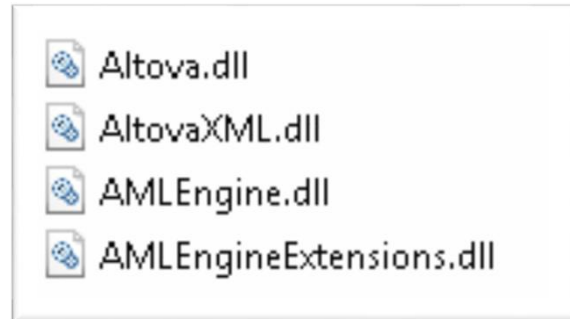
04 ExampleApplication_TutorialForSplitMergeAndValidate

contains an example application demonstrating how to split and merge CAEX files

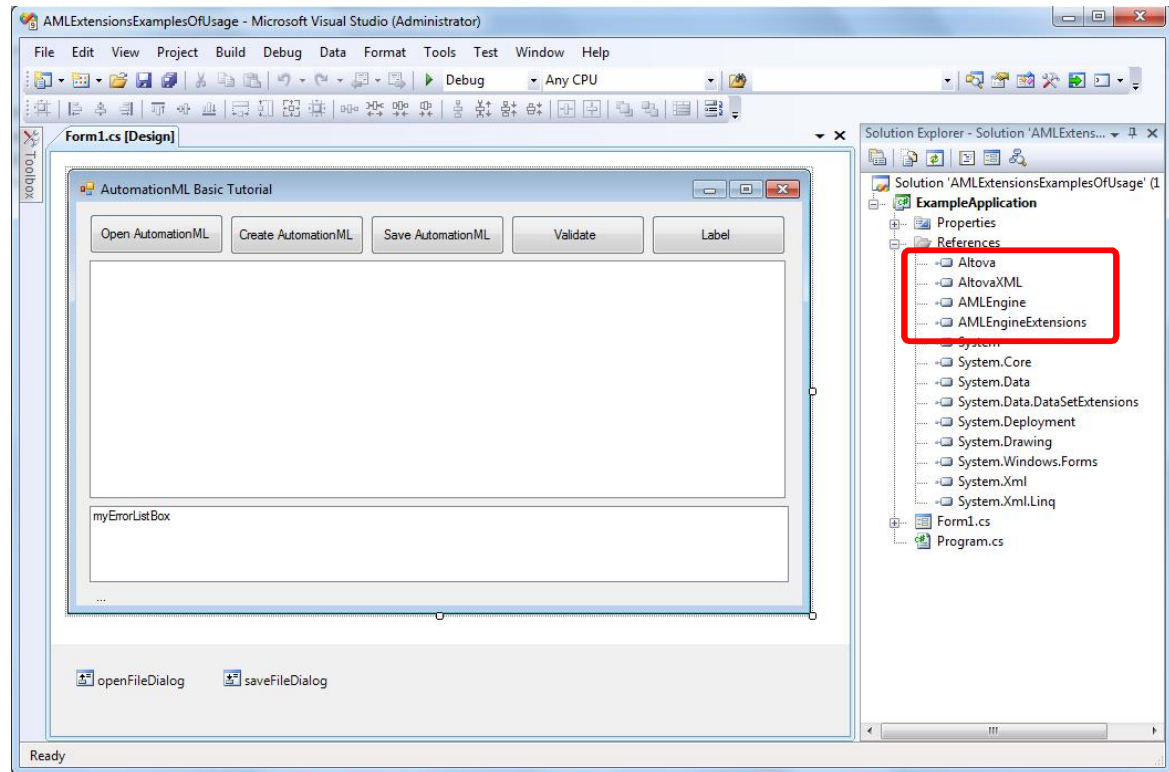
05 ExampleFiles

Contains a set of example CAEX files with different kinds of typical errors in order to demonstrate the validation capabilities of the AutomationML Engine

How to embed the AutomationML engine ... in your own C# project



- Bind these files into your .Net project



Basic use cases

Loading the CAEX namespace

```
using CAEX_ClassModel;
```

Create a new CAEX document

```
CAEXDocument myDoc = CAEXDocument.New_CAEXDocument();
```

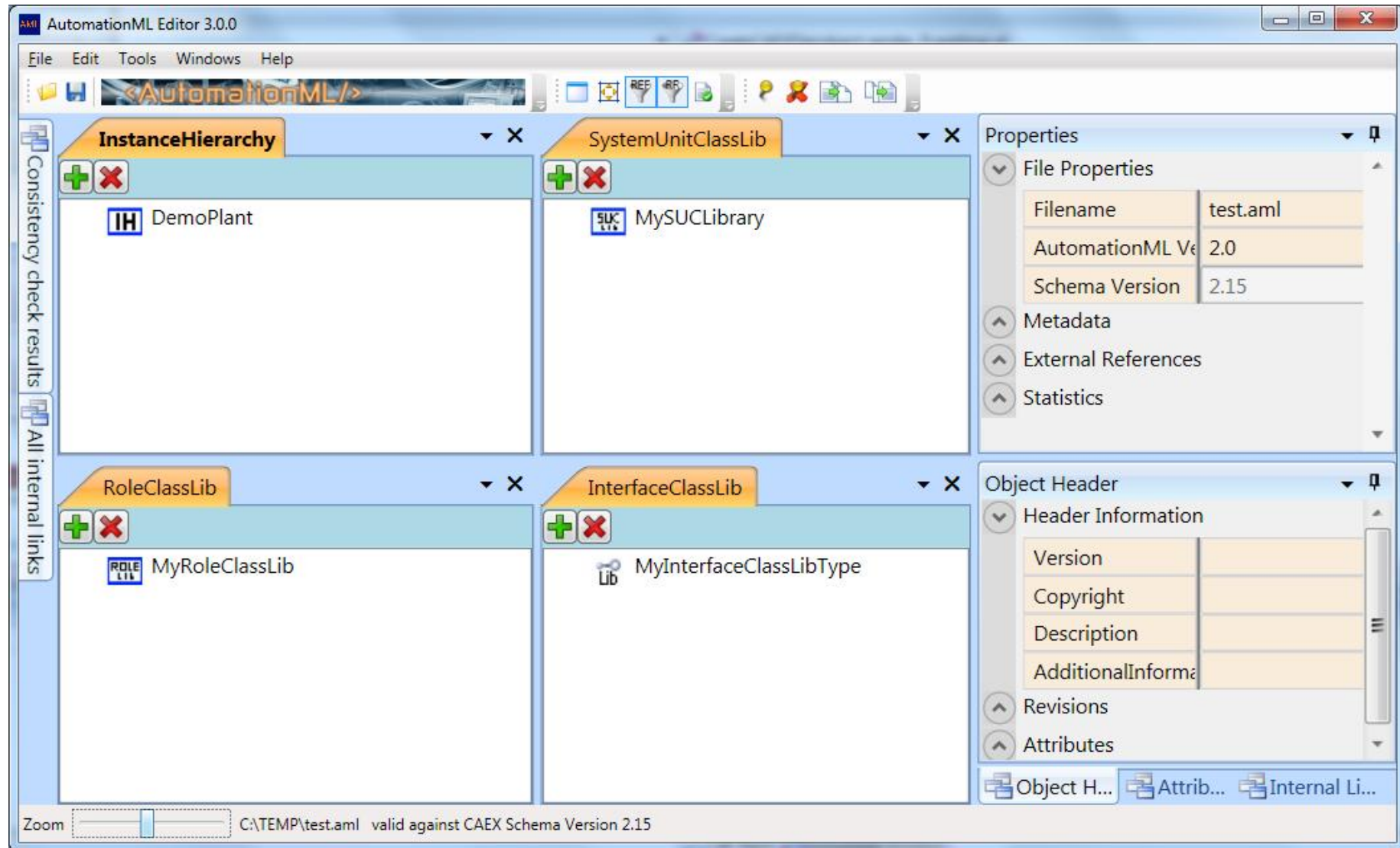
Load a CAEX document

```
myDoc = CAEXDocument.LoadFromFile(@"c:\temp\test.aml");
```

Save a CAEX document

```
myDoc.SaveToFile(@"c:\temp\test.aml", true);
```

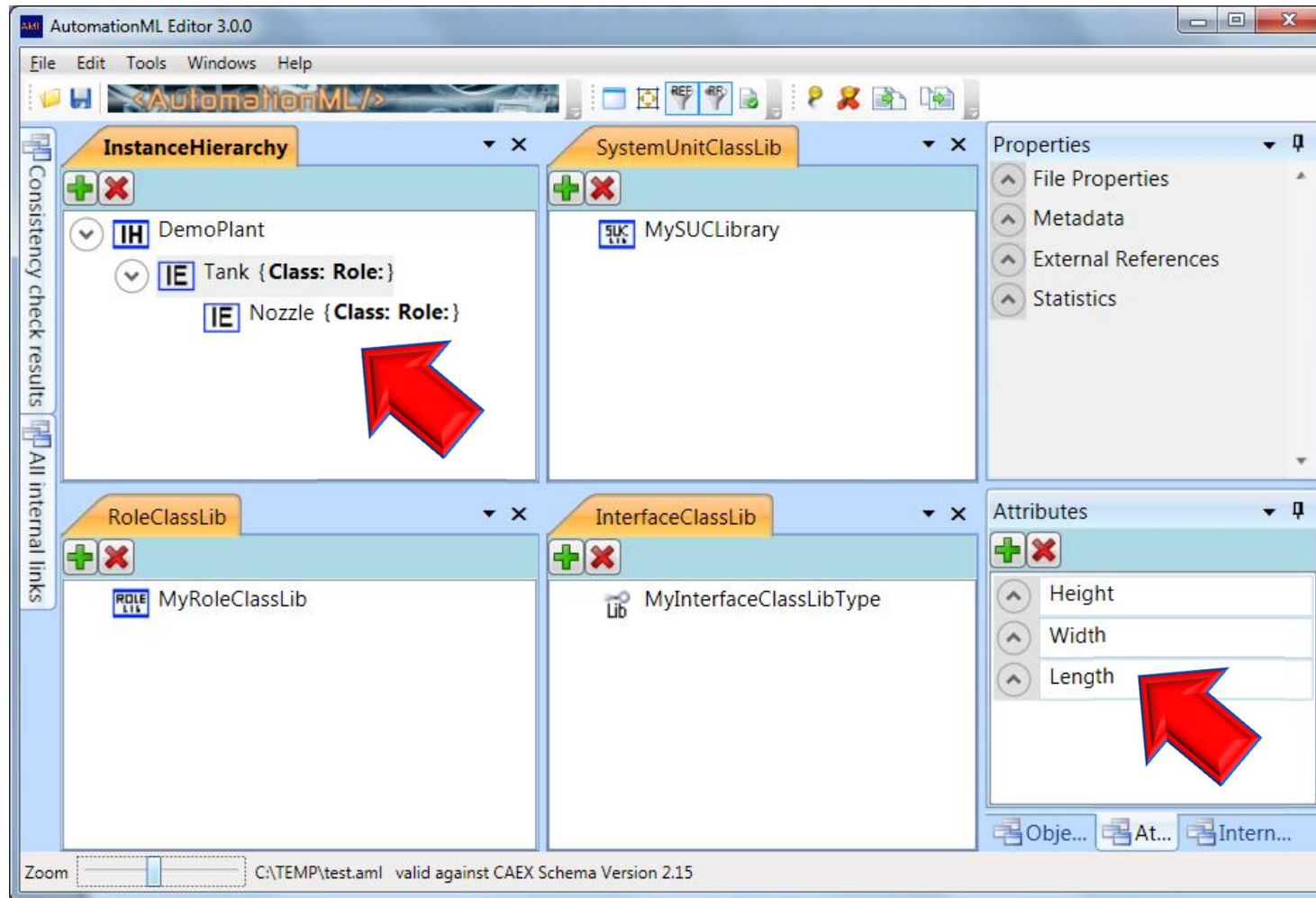
Creating CAEX Hierarchies



Creating CAEX Hierarchies

```
//variable declaration  
CAEXFileType myCAEXFile = myDoc.CAEXFile;  
InstanceHierarchyType IH;  
SystemUnitClassLibType SUCL;  
RoleClassLibType RCL;  
InterfaceClassLibType ICL;
```

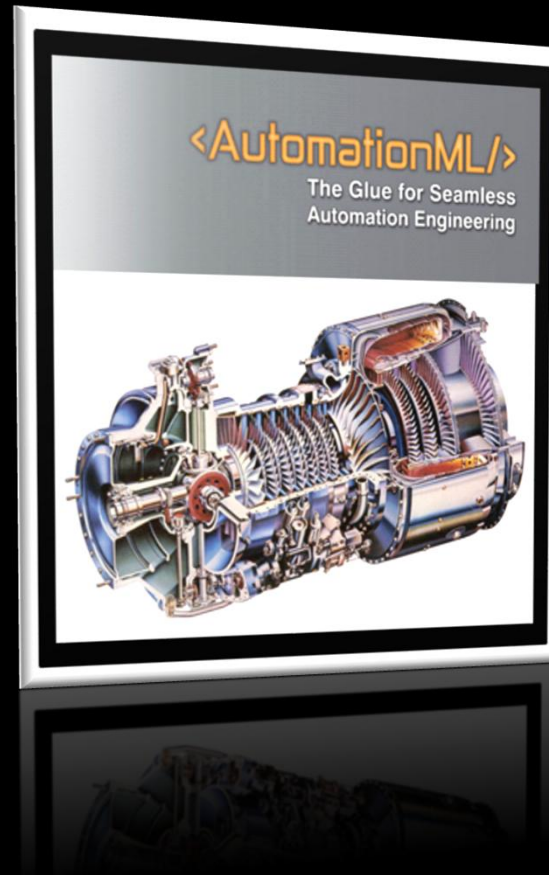

Creating InternalElements



Creating InternalElements

```
//create internal element
InternalElementType IE1, IE2;
IE1 = IH.New_InternalElement("Tank");
IE1.New_Attribute("Length");
IE1.New_Attribute("Width");
IE1.New_Attribute("Height");
//create child internal element
IE2 = IE1.New_InternalElement("Nozzle");
```

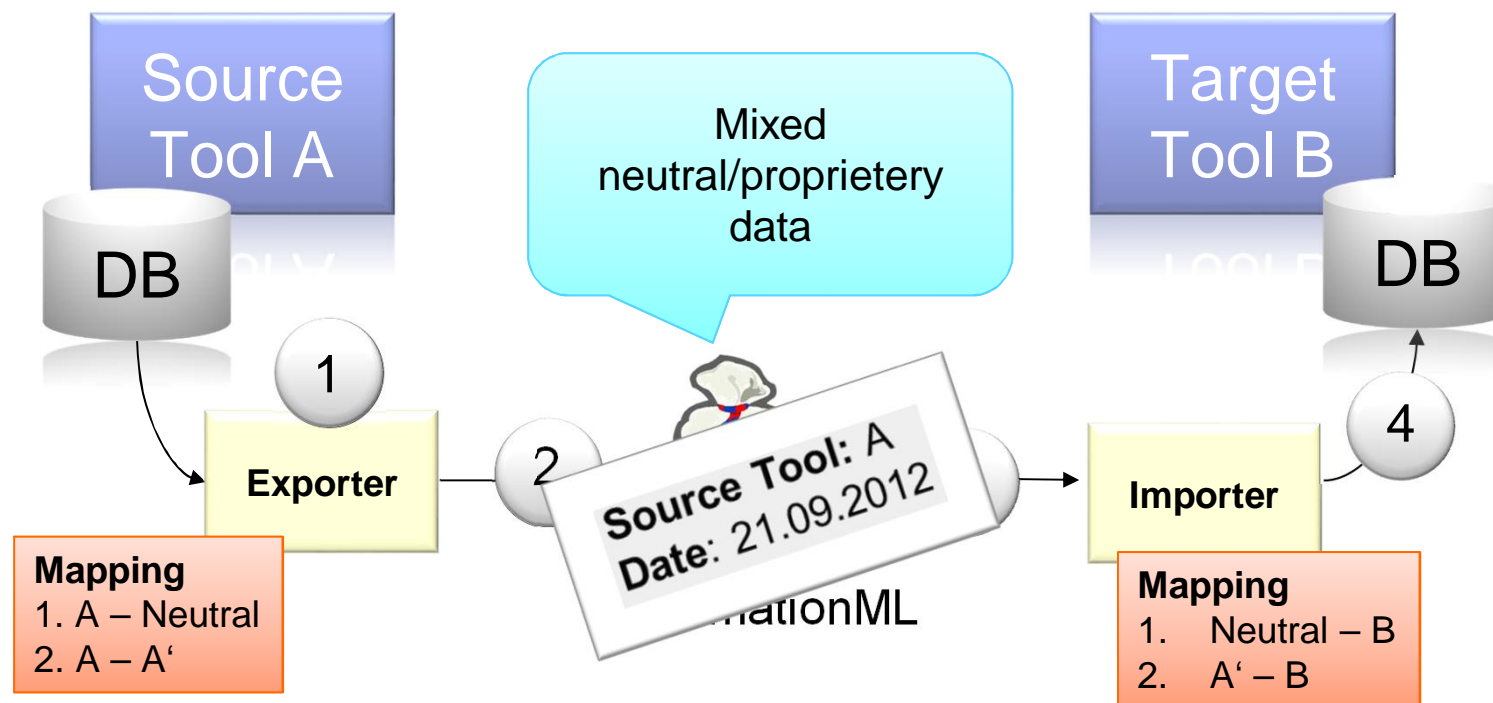
AutomationML programming



Extended use cases

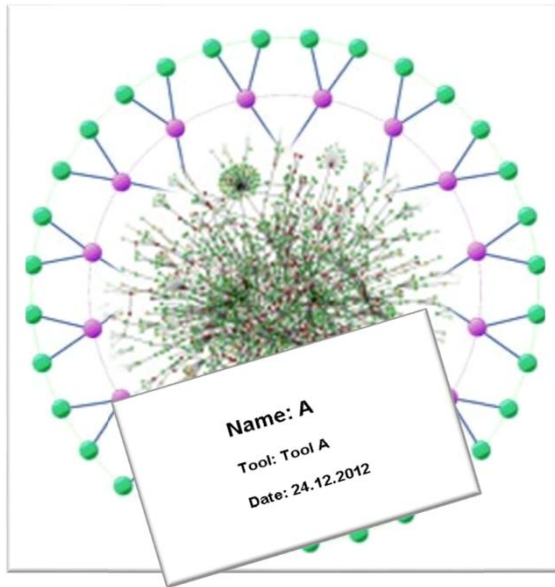
How to deal with semantic variety

Source Tool: A
Date: 21.09.2012



The trick

Label your CAEX file!



- we put a label on the „neutral file“.

XML tag name	Example
WriterName	“a Source Tool”
WriterID	“a Source Tool”
WriterVendor	“a company”
WriterVendorURL	“www.acompany.com”
WriterVersion	“1.0”
WriterRelease	“1.0.1”
LastWritingDateTime	“2012-02-01T16:23:00”
WriterProjectTitle	“eCarproduction”
WriterProjectID	“eCarproduction_LinePLC.prj”

Create a Label

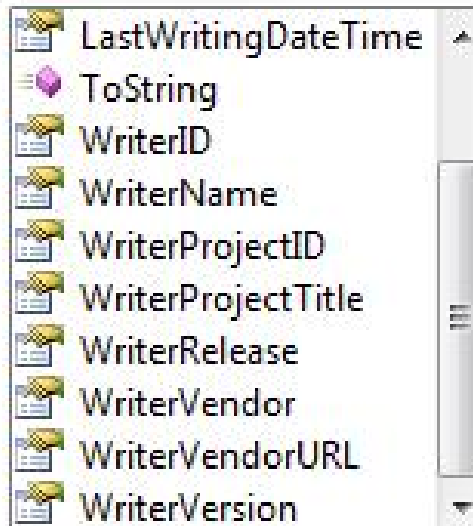
```
MetaInformation m = new MetaInformation();
```

```
myDoc.CAEXFile.SetMetaInformation(m);
```


Read a label

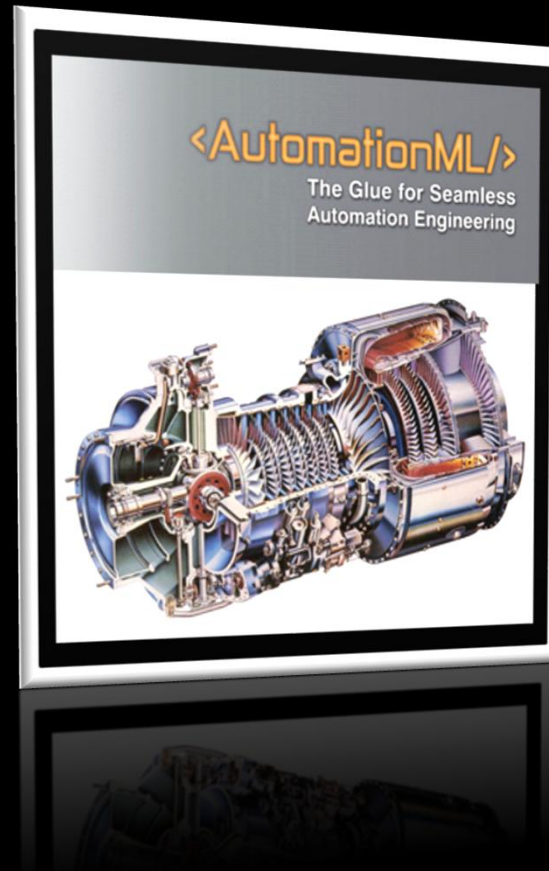
```
MetaInformation m = myDoc.CAEXFile.GetMetaInformation().First();
```

m.



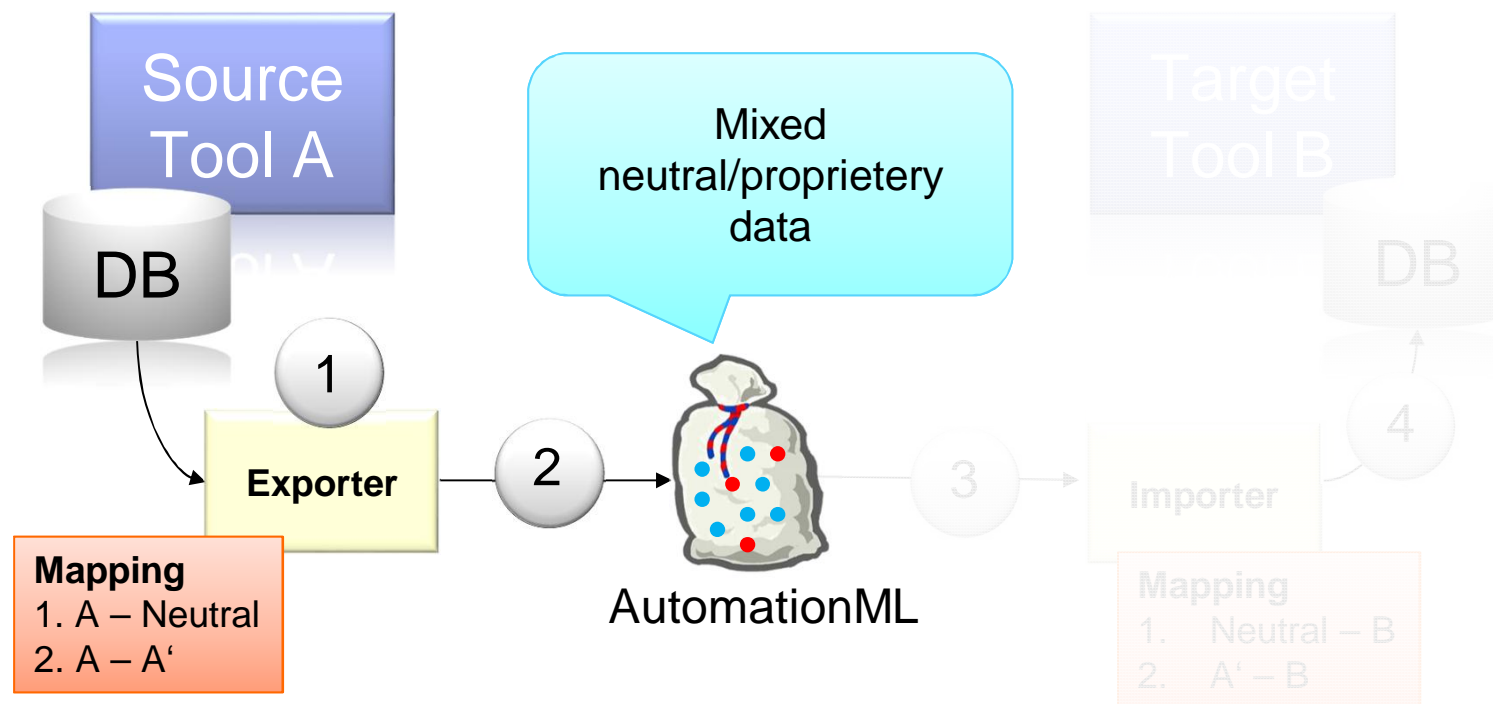
- LastWritingDateTime
- ToString
- WriterID
- WriterName
- WriterProjectID
- WriterProjectTitle
- WriterRelease
- WriterVendor
- WriterVendorURL
- WriterVersion

AutomationML programming

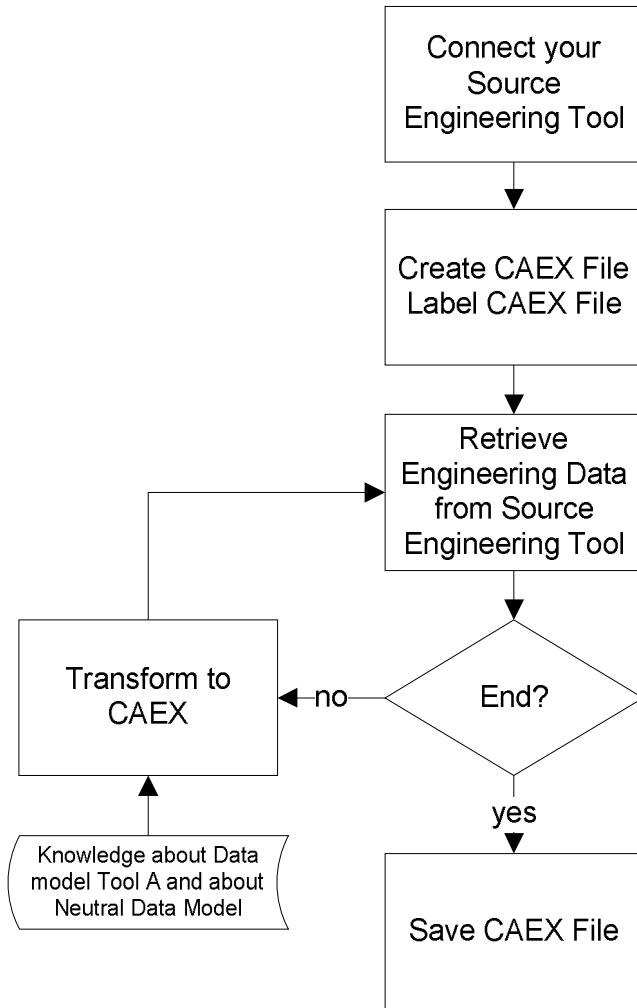


Basic design of an CAEX Exporter

Exporter programming



How to export AutomationML Files

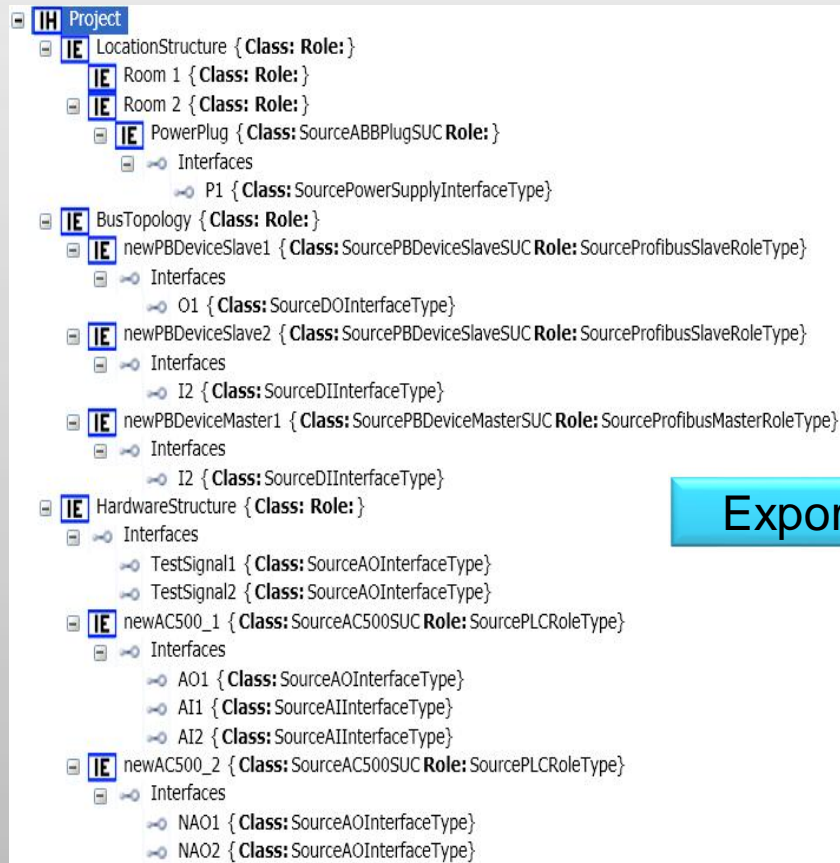


Result: a CAEX file that contains 100% neutralized data syntax, but 100% proprietary data semantics. or 50/50%.

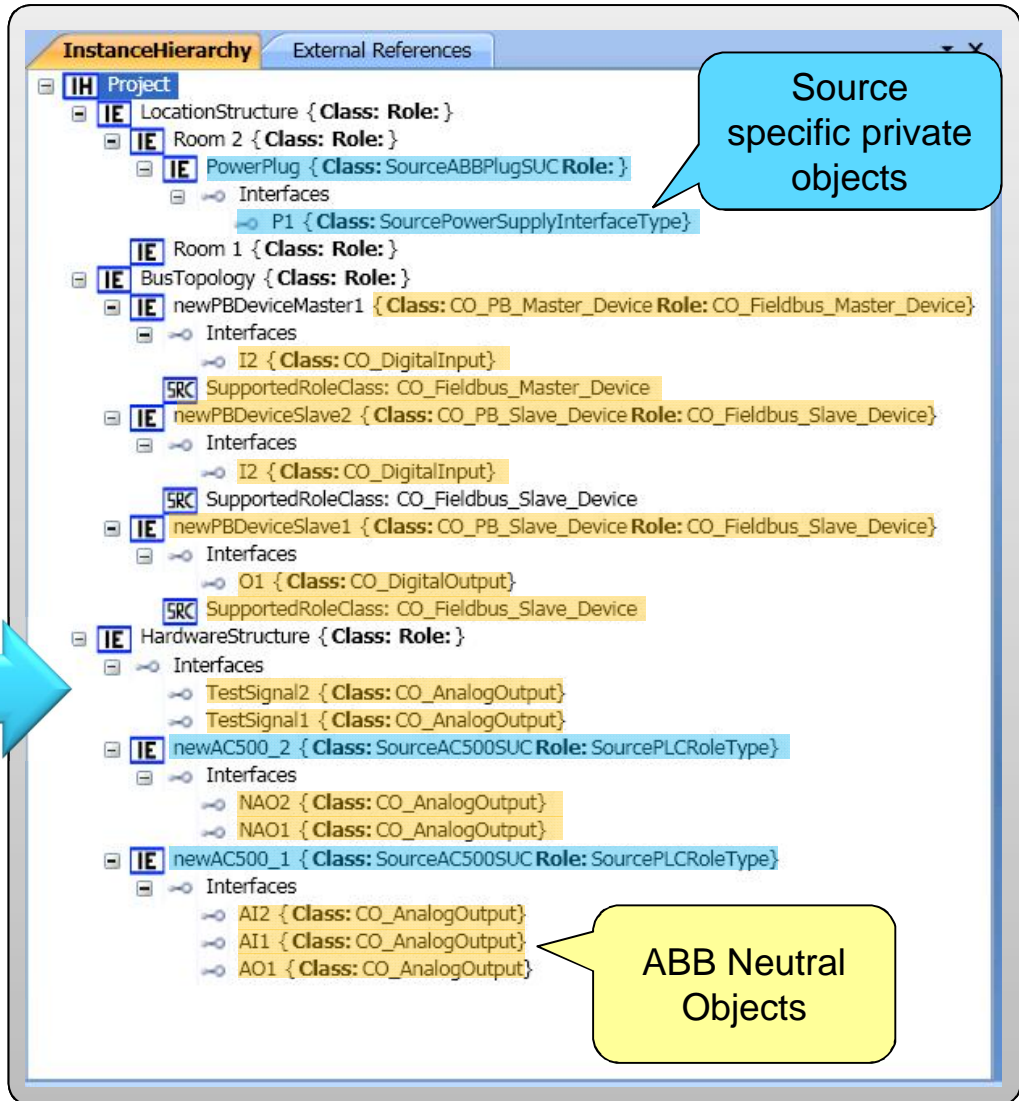


Example

Source Tools project structure



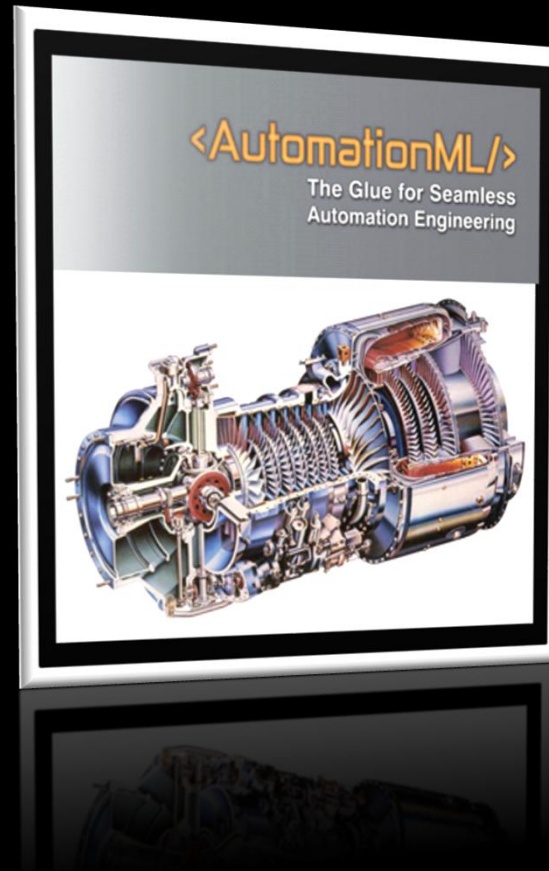
Export



Source specific private objects

ABB Neutral Objects

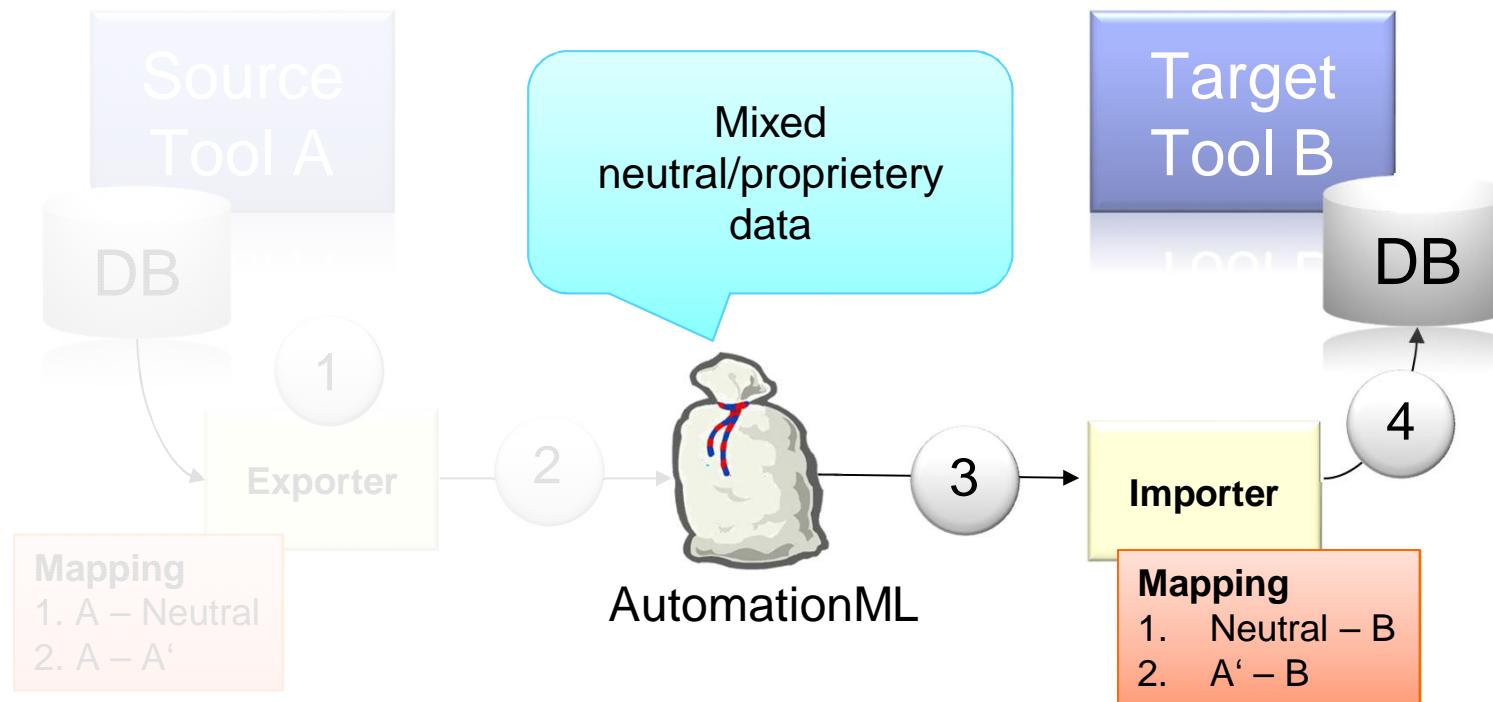
AutomationML programming



How to program CAEX Importer

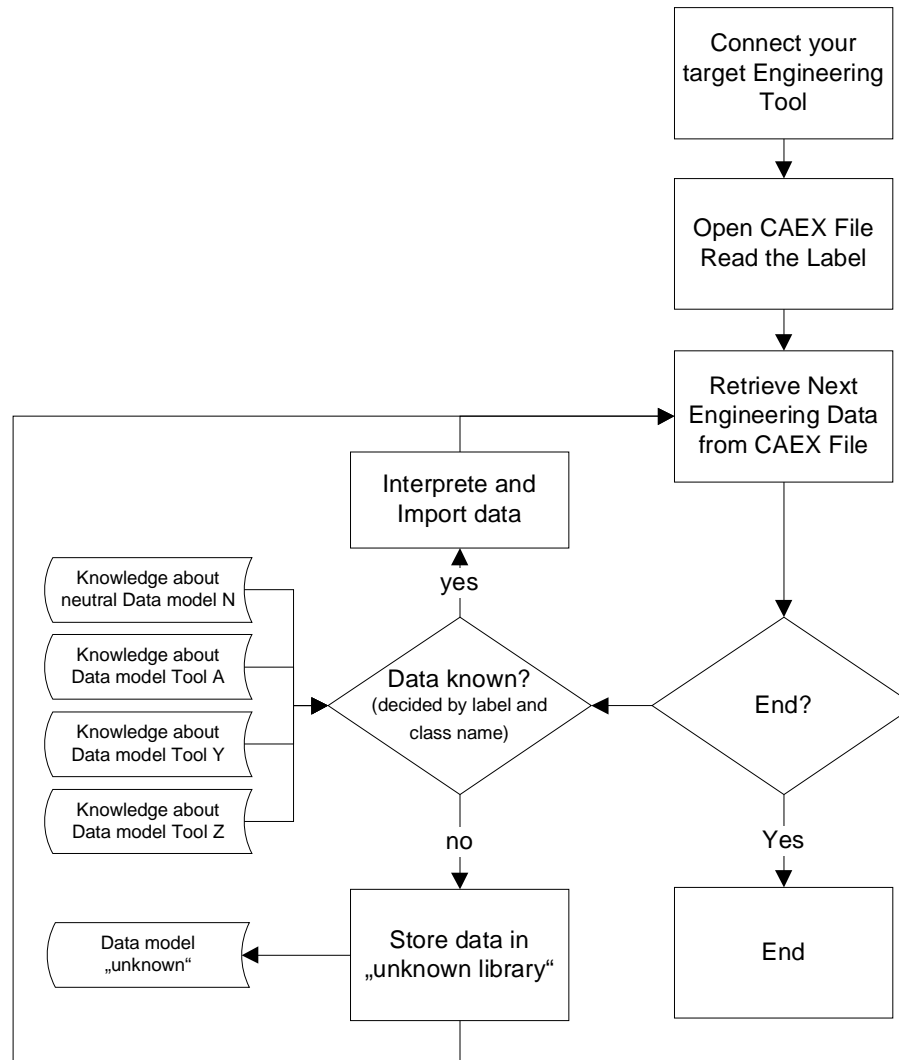
Importer programming

Scenario: one shot import



How to import AutomationML Files

It is (often much more) easier to implement a transformation routine for known source engineering tools than standardize a common engineering world model.



Example 1

The screenshot displays the Visual Studio IDE with several windows open, illustrating the structure of ABB Neutral Objects and Source specific private objects.

Instance Hierarchy Window:

- Project**
 - LocationStructure {Class: Role:}**
 - Room 2 {Class: Role:}**
 - PowerPlug {Class: SourceABBPlugSUC Role:}**
 - Interfaces**
 - P1 {Class: SourcePowerSupplyInterfaceType}**
 - Room 1 {Class: Role:}**
 - BusTopology {Class: Role:}**
 - newPBDeviceMaster1 {Class: CO_PB_Master_Device Role: CO_Fieldbus_Master_Device}**
 - Interfaces**
 - I2 {Class: CO_DigitalInput}**
 - SupportedRoleClass: CO_Fieldbus_Master_Device**
 - newPBDeviceSlave2 {Class: CO_PB_Slave_Device Role: CO_Fieldbus_Slave_Device}**
 - Interfaces**
 - I2 {Class: CO_DigitalInput}**
 - SupportedRoleClass: CO_Fieldbus_Slave_Device**
 - newPBDeviceSlave1 {Class: CO_PB_Slave_Device Role: CO_Fieldbus_Slave_Device}**
 - Interfaces**
 - O1 {Class: CO_DigitalOutput}**
 - SupportedRoleClass: CO_Fieldbus_Slave_Device**
 - HardwareStructure {Class: Role:}**
 - Interfaces**
 - TestSignal2 {Class: CO_AnalogOutput}**
 - TestSignal1 {Class: CO_AnalogOutput}**
 - newAC500_2 {Class: SourceAC500SUC Role: SourcePLCRoleType}**
 - Interfaces**
 - NAO2 {Class: CO_AnalogOutput}**
 - NAO1 {Class: CO_AnalogOutput}**
 - newAC500_1 {Class: SourceAC500SUC Role: SourcePLCRoleType}**
 - Interfaces**
 - AI2 {Class: CO_AnalogOutput}**
 - AI1 {Class: CO_AnalogOutput}**
 - AO1 {Class: CO_AnalogOutput}**

SystemUnitClassLib Window:

- UnknownSystemUnitClassLib**
 - SourceABBPlugSUC {Class:}**
 - SourceAC500SUC {Class:}**
- CollaborationObjectLib**
 - CO_Fieldbus_Slave_Device {Class:}**
 - CO_PB_Slave_Device {Class: CO_Fieldbus_Slave_Device}**
 - SupportedRoleClass: CO_Fieldbus_Slave_Device**
 - CO_Fieldbus_Master_Device {Class:}**
 - CO_PB_Master_Device {Class: CO_Fieldbus_Master_Device}**
 - SupportedRoleClass: CO_Fieldbus_Master_Device**

InterfaceClassLib Window:

- UnknownInterfaceClassLib**
 - SourcePowerSupplyInterfaceType {Class:}**
- CollaborationObjectInterfaceLib**
 - CO_Signal {Class:}**
 - CO_DigitalSignal {Class: CO_Signal}**
 - CO_DigitalOutput {Class: CO_DigitalSignal}**
 - CO_DigitalInput {Class: CO_DigitalSignal}**
 - CO_AnalogSignal {Class: CO_Signal}**
 - CO_AnalogOutput {Class: CO_AnalogSignal}**

RoleClassLib Window:

- CollaborationObjectStandardRoleLib**
 - CO_Fieldbus_Slave_Device {Class:}**
 - CO_PB_Slave_Device {Class: CO_Fieldbus_Slave_Device}**
 - CO_Fieldbus_Master_Device {Class:}**
 - CO_PB_Master_Device {Class: CO_Fieldbus_Master_Device}**
- UnknownRoleClassLib**
 - SourcePLCRoleType {Class:}**

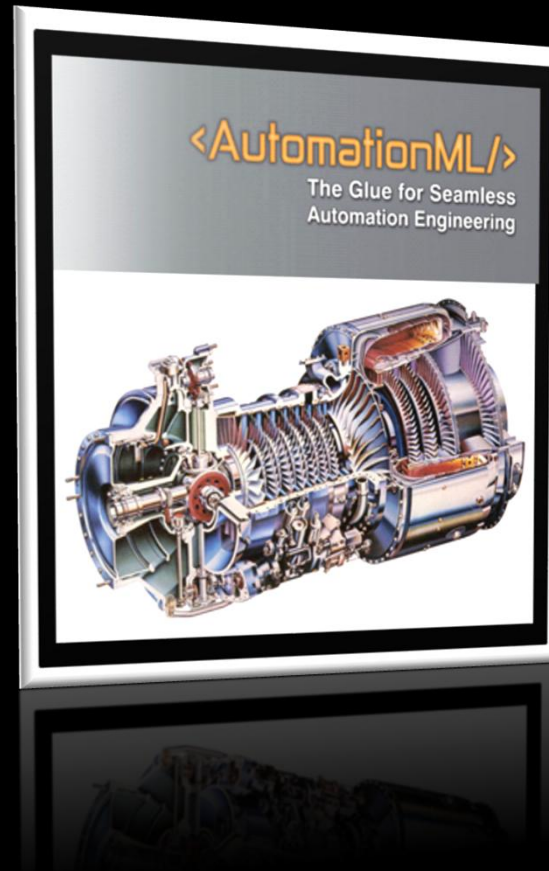
CALLBACK Scene View Window:

- COLLADA Scene View**

Annotations:

- Source specific private objects:** A blue callout bubble points to the **Source specific private objects** (e.g., **SourcePowerSupplyInterfaceType**, **SourceAC500SUC**).
- ABB Neutral Objects:** A yellow callout bubble points to the **ABB Neutral Objects** (e.g., **CO_DigitalInput**, **CO_DigitalOutput**).

AutomationML programming



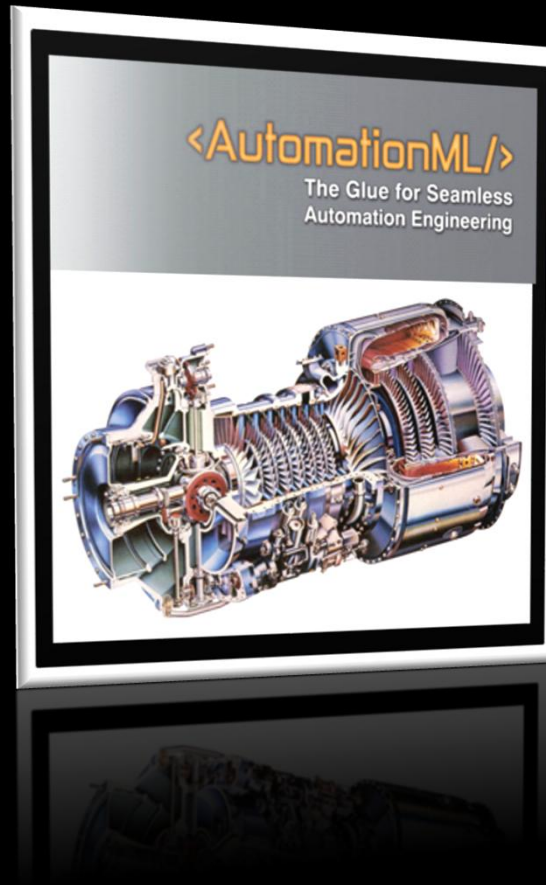
Validation of CAEX files

Rules

Extract:

- Existence of the AutomationML Version tag “AutomationML-Version 2.0”
- Existence of vendor specific meta data (the AutomationML label)
- Existence of CAEX schema file and correct naming of that file
“CAEX_ClassModel_V2.15.xsd”
- Check existence of external CAEX files referenced with “ExternalReferences”
- Check existence of external data files (Collada, PLCopen XML)
- Check: alias's shall have different names
- Check: existence of external CAEX files (splitting)
- Check: all InternalElements and Interface-Instances must have a unique ID
- Check: Attributes shall have different names among siblings
- Check: mirror objects must have a unique ID

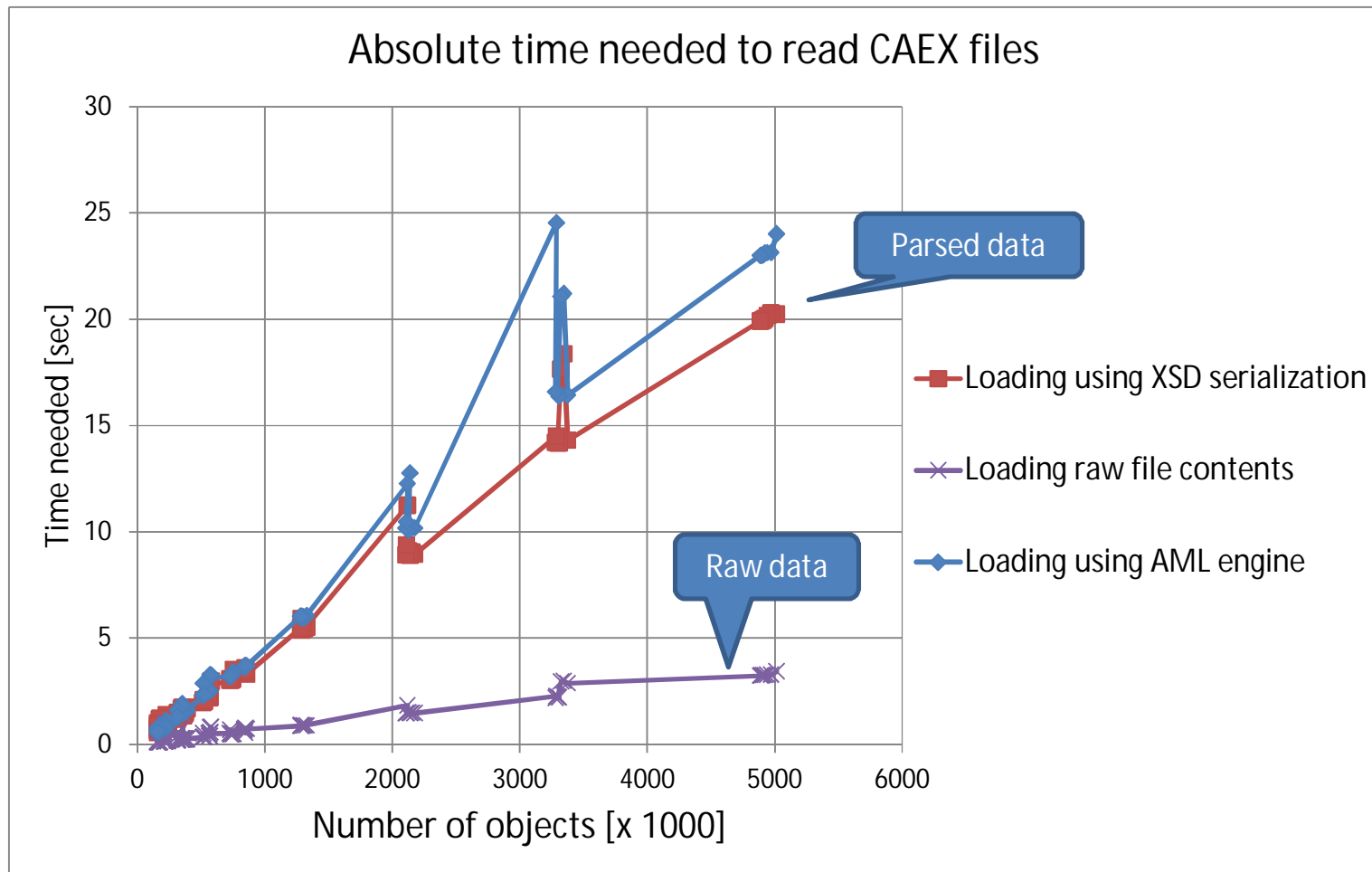
AutomationML programming



Performance

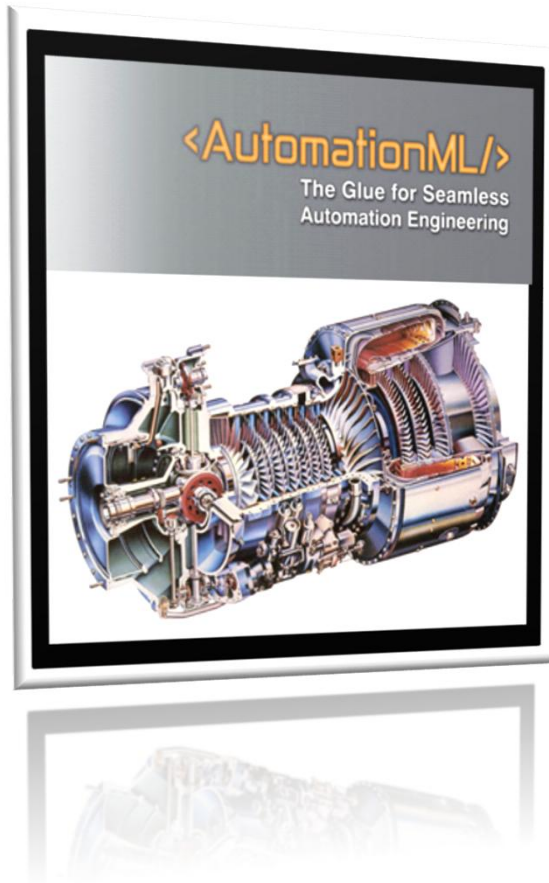
Performance measurement

How fast is the AutomationML engine?



Summary

What have we learned in this workshop?



Programming CAEX

- Simple
- Fast

AutomationML engine

- full support of newest AutomationML technologies
- Qualification niveau to program CAEX: student
- don't wait for semantic standards!

Published

- www.automationml.org
- License: free under LGPL license

Power and productivity
for a better world™

