Critical System

Digital Twin

Autor
Abdullah Taher Saadoon AL-Musawi
DNI: A11518169
Email: abdullahtaher@correo.ugr.es

Escuela Ténica Superior de Ingenierías Informática y de Telecomunicación

—

Granada, mayo de 2020

Contents

1. The Digital Twin

Digital twins refer to virtual replicas of physical objects that, inter alia, enable to monitor, visualize, and predict states of cyber-physical systems (CPSs)
A digital twin is an integrated [. . .] simulation of a [. . .] system that uses the best available physical models, sensor updates, [. . .] etc., to mirror the life of its [. . .] flying twin. Shafto et al

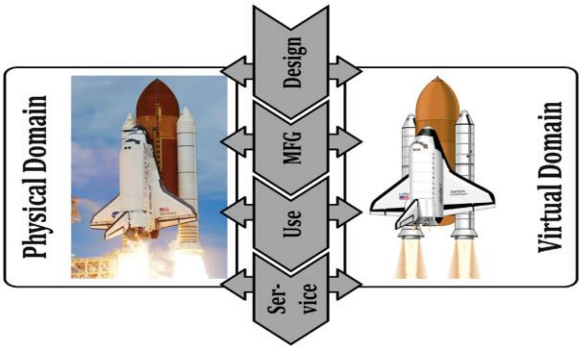
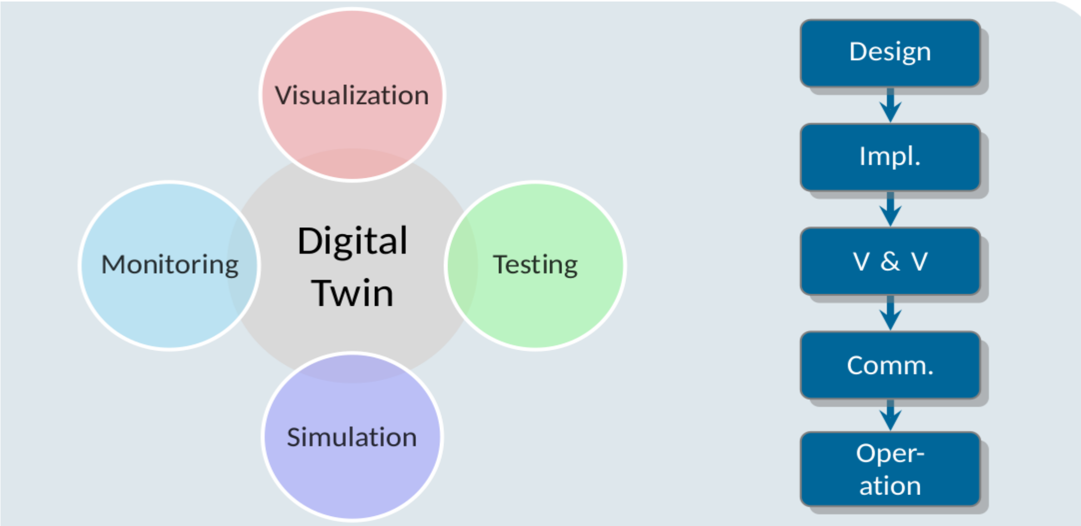


Figure: The vision according to

(a) Nuclear power plant © AlMare, CC BY-SA 3.0 (b) Industrial Robots © Mixabest, CC BY-SA 3.0 (c) Tesla Model S © raneko, CC BY 2.0

1.2 Use Cases of the Digital Twin Concept



Let’s have a look at some of the use cases of Digital Twin

1.2.1- Healthcare

In healthcare digital twins of particular patient organs allow doctors to test different care delivery approaches and prevent conditions that are years apart and enable patient-specific surgery training to prepare for complex invasive procedures. With insight created by digital twins, healthcare organizations can create innovative practices to correct the problem and minimize the possibility of any kind of risk. Also, real-time epidemiology data can help hospital staffers to track where infections agents may exists and who could be at risk by contacts.

1.2.2- Utilities

In utilities, the robust system enables efficient management of a wide range of document, including 3D models, 2D process schematics, as-built documents, instrumentation data and other business and critical Engineering & Design information. Here Digital Twin real-time data which This allows project manager at oil and gas companies to achieve a real ‘as-designed,’ ‘as-built,’ ‘as-operated’ visibility into their fleet of globally distributed assets, enabling them to quickly and efficiently re-plan and reposition the fleet to access new opportunities globally.

1.2.3- Disaster Management

Earth’s climate is now changing faster than at any point in the history of modern civilization, primarily as a result of human activities. Global climate change has already resulted in a wide range of impacts across every region of the country and many sectors of the economy that are expected to grow in the coming decades. Fires, flood, and drought have become normal now. In such a scenario, Digital Twin can be used to build smarter infrastructure like dams, utility networks, emergency response plans, and zoning.

1.2.4- Insurance

For insuring anything from any kind of risk it is important to understand the location of the risk. For example, insure a famous historical monument from risk of terrorist attack, a Digital Twin model can help to understand that which is the area from where terrorists can easily enter in the monument. Once the location is known it is easy to take protective measures. Another use case of Digital Twin in Insurance can be use of the technology while building any infrastructure, where it can help to access risks in advance.

1.2.5- Smart Cities

Digital Twin can help cities to become more environmentally, economically and socially sustainable. It enable users to create models that guide their future plans and help provide solution to the complex issues that cities face. In case of any disaster like flood, Digital Twin provides useful information in real-time, like which areas are flooded, which infrastructure will be closed down, which hospitals could be affected and thus allowing city managers to take immediate action.

1.3 Security-specific Use Cases of the Concept

1.3.1- Intrusion Detection

- Knowledge-based
- Behavior-specification-based
- Process knowledge

1.3.2- Detecting Misconfigurations

- Manipulation by attacker
- Detect unknown devices

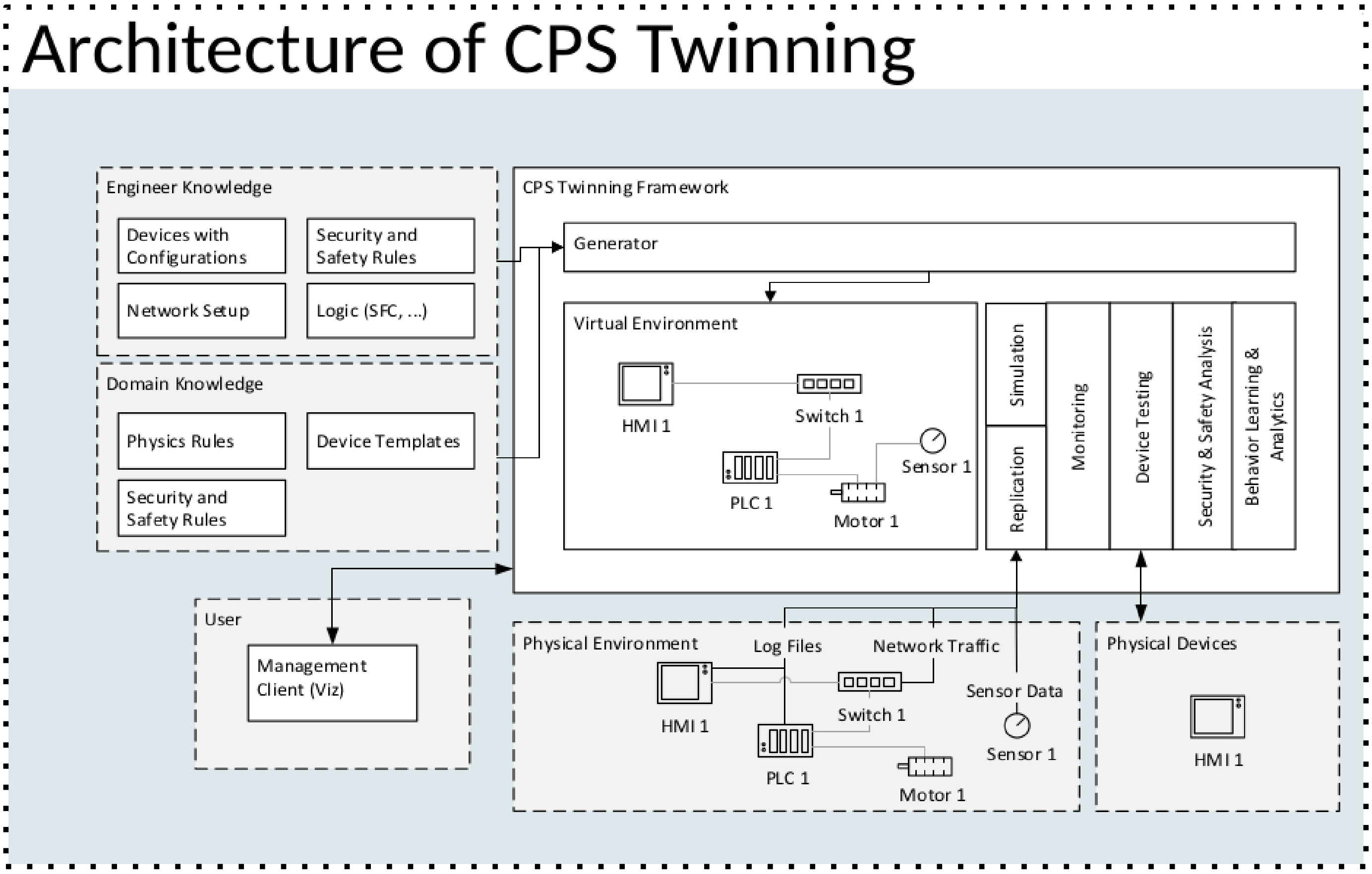
1.3.3- Penetration Testing

- No interference with live system
- No test environment required

2. CPS Twinning

CPS Twinning is a framework for generating and executing digital twins that mirror cyber-physical systems (CPSs). This framework allows to automatically generate virtual environments for digital twins completely from specification. Ideally, artifacts that specify the correct behavior of a CPS have already been created during the production system engineering (PSE) process and will be maintained throughout the entire lifecycle. Standardized engineering data exchange formats, such as AutomationML (AML), may facilitate this process.

On top of CPS Twinning, multiple use cases can be implemented, ranging from behavior-specification-based intrusion detection to behavior learning & analytics.

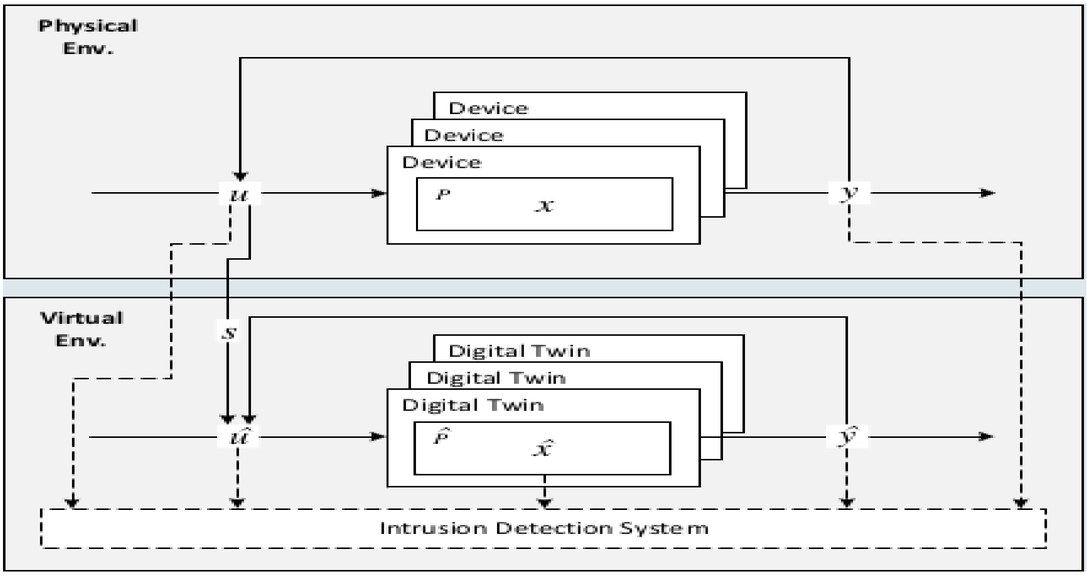


3. State Replication

3.1 A Passive State Replication Approach

A FSM, $P := (X, x_0, U, Y, \delta, \lambda)$

- **X is the finite set of states**
- **$x_0 = X$ is the initial state**
- **U is the finite set of inputs**
- **Y is the finite set of outputs**
- **δ is the transition function**
- **λ is the output function**



We expect that P = P̂

Thus, $\delta(x, u) = \hat{\delta}(\hat{x}, \hat{u}) \Leftrightarrow x' = \hat{x}'$, provided that $(x = \hat{x}) \wedge (u = \hat{u})$.

S, denotes the set of stimuli

$$S := \{ z \in \hat{U} \mid z \in U \wedge z \notin Y^* \}$$

Each digital twin should produce $\hat{y} \in \hat{Y}$ by itself.

Use specification of CPS to identify stimuli

Let $f\colon U^* \cup Y^* \rightharpoonup S^*$ be a partial function, then I is defined as follows:

$$I := \{ j \in U^* \cup Y^* \mid f(j) \in S^* \}.$$

Replicate stimuli

Next, $j \in U^* \cup Y^*$ will be observed and checked whether $j \in I$.

Since $j \in I \Leftrightarrow f(j)\downarrow, s \in S^*$, the value of f of j, is fed to the respective digital twin.

Hence, $\hat{\delta}(\hat{x}, s) = \hat{x}'$.

Example:-

- **Conveyor belt**
- **HMI & PLC digital twins exist**
- **Communication via Modbus TCP/IP**
- **Definition of f**
- **AutomationML [2]**

```
<InternalElement Name="LogicalNetwork" ID="c51...">
  <InternalElement Name="ModbusRequests" ID="ce1...">
    <InternalElement Name="StartConveyorBeltModbusReadRequest" ID="0e5...">
      <Attribute Name="functionCode" AttributeDataType="xs:integer">
        <Value>3</Value>
      </Attribute>
      ...
      <InternalLink Name="HMI1 StartConveyorBelt - PLC1 Modbus 400001" RefPartner-SideA="{068...}:StartConveyorBelt" RefPartnerSideB="{29b...}:1" />
      <RoleRequirements RefBaseRoleClass-Path="/ModbusReadHoldingRegisters" />
    </InternalElement>
    ...
```

4. CPS Installation

CPS Twinning depends on Mininet-WiFi, MatIEC and CPS State Replication.

Mininet-WiFi

Mininet-WiFi is a fork of Mininet (http://mininet.org/) which allows the using of both WiFi Stations and Access Points. Mininet-WiFi only add wifi features and you can work with it like you were working with Mininet.

**** Things to keep in mind when working with Mininet-WiFi**

- You can use any wireless network tools (e.g. iw, iwconfig, wpa_supplicant,etc)

- Please consider computer network troubleshooting steps to solve issues before making questions in the mailing list (e.g. is the station associated with ap? Is the OpenFlow rule working correctly? etc)
- Do you need help? Be careful with questions in the mailing list and please providing as much information you can.

** ***User Manual/(we can find it in the folder of the practice)***

Installation
step 1: $ sudo apt-get install git
step 2: $ git clone https://github.com/intrig-unicamp/mininet-wifi
step 3: $ cd mininet-wifi
step 4: $ sudo util/install.sh -Wlnfv

MatIEC

First, install the MatIEC dependencies flex and bison. For example:

```
$ sudo apt-get update
$ sudo apt-get install flex
$ sudo apt-get install bison
```

Then, clone the MatIEC repository and build the two transcompilers:

```
$ hg clone ssh://hg@bitbucket.org/mjsousa/matiec
$ cd matiec
$ autoreconf -i
$ ./configure
$ make
```

After that, set the following environment variables:

```
$ export MATIEC_INCLUDE_PATH=/home/<user>/matiec/lib
$ export MATIEC_C_INCLUDE_PATH=/home/<user>/matiec/lib/C
$ export PATH=/home/<user>/matiec:$PATH
```

CPS Twinning

Finally, clone this repository and install CPS Twinning:

```
$ git clone https://github.com/sbaresearch/cps-twinning.git
$ cd cps-twinning
$ virtualenv env
$ source env/bin/activate
(env)$ pip install -r requirements.txt
(env)$ pip install .
```

Usage

Now, to start CPS Twinning, run make twinning and can be initiated by executing twinning <path_to_aml>. An exemplary specification can be found at misc/specification/CandyFactory.aml.

Note that this project is only a proof of concept. As a consequence, there are currently many areas that need improvements. In particular, the functionality of the AutomationML parser is currently limited and may require manual adjustments.

5. Automation Project Configuration

A very frequently occurring task within the planning process of production and automation systems is the exchange of automation project configuration information of automation system devices between ECAD and engineering systems. This information includes a wide range of diverse hardware modules and systems, like PLCs, decentral IO-devices and drives. To avoid manual transfer of the engineering data between the participating systems, ECAD and engineering systems need an interface for sharing this information. This information will not only include all necessary hardware modules and interconnections, yet also additional logical and structural information of the system, which are provided at an early stage of the planning process and have to be preserved through the whole planning process.
Using this automatic data transfer errors on manual copying data can be avoided. Also, for changes of the system, the reaction time can be reduced to a large extend.

6. notes regarding exchange of configuration data for drives

Systems containing drives may comprise different aspects other than electrical configuration. For systems like robots or tooling machines the machine concept may start with functional view, which includes also dynamic, mechanical and geometrical aspects. Tools used to define this machine concept at an early stage of the machine design provide information on the functional behaviour of the intended machine. Starting from this initial functional behaviour the mechanical and electrical design of the system will be derived, as well as the programming aspects in a later stage of system implementation. Based on the functional requirement of the system or machine, the different electrical components can be derived, which are needed to build the machine configuration. Dynamic and mechanical aspects also provide information for the dimensioning of the electrical components, which can be used to select fitting components during the electrical design of the system or machine to be build. While the mechanical aspects are not relevant for the electrical design 1 , the information concerning the drive is essential for the electrical design as well as for the programming aspects of PLC tools.
While a machine concept design tool, ECAD tools and tools for engineering and PLC programming have different views of the automation system information, they share common information, which must be kept consistent among different tools. This does not only apply to the initial configuration of the system, it also applies to changes during system engineering, refitting the system, as well as simulation and diagnosis of the system.
Other tools may also be included within the data transfer or use the information exchanged, as shown in Figure. Nevertheless, within this document focus on ECAD and PLC programming tools.

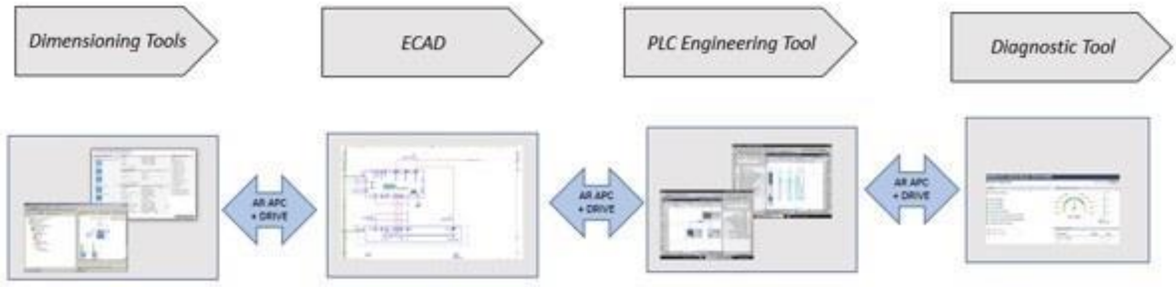
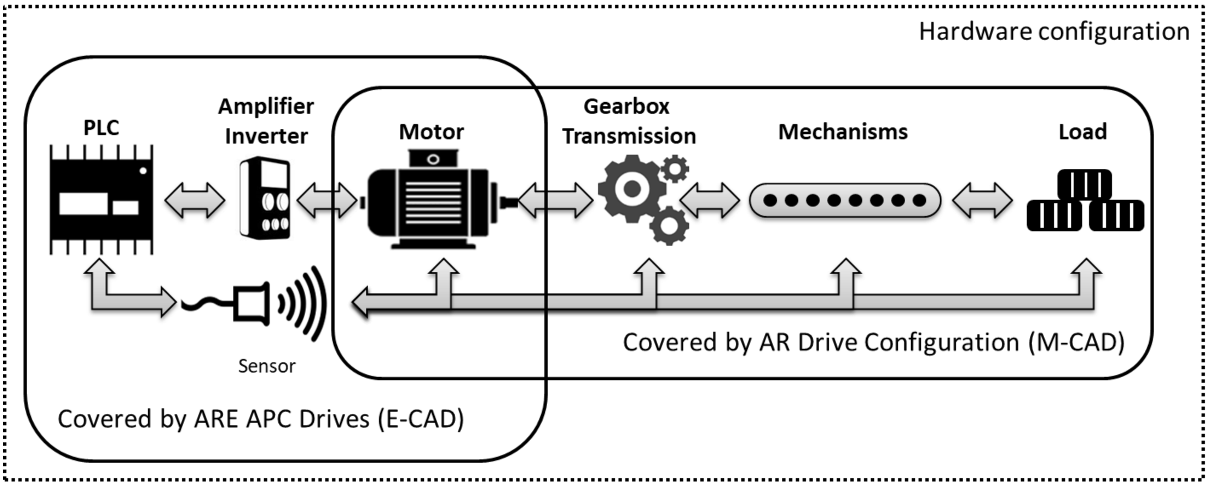


Figure 2 shows the different aspects of drive configurations. In this document the hardware parts and the electrical wiring aspects are covered. The mechanical and functional aspects are covered in the Application Recommendation Drive for MCAD (AR Drive MCAD).



6.1 Data exchange workflow

Usually the configuration of a system or machine using drives starts with a functional view of system which also defines the dynamic aspects. This design process can be supported by a machine concept design tool. Based on this information the mechanical and electrical configuration of the systems starts, where the corresponding electrical elements will be chosen to fulfil the functional and dynamic aspects of the system. This process may also be supported by a hardware selection tool.

6.2 Recommended workflow

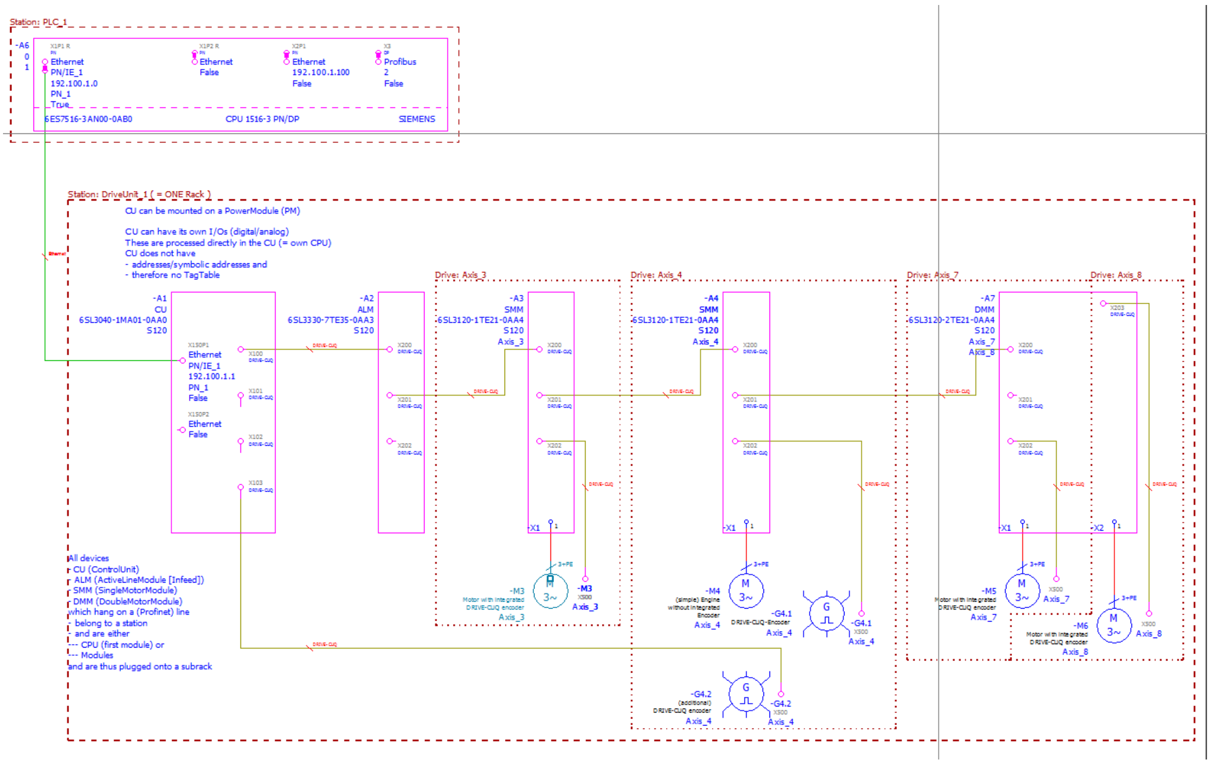
According to the described criteria in most cases the following workflow is established.

- Set up a functional view of the drive system which defines its functional and dynamic aspects.
- Select electrical hardware modules, which meet the derived requirements.
- Set up the machine configuration and networking with a configuration tool.
- Import this configuration to ECAD tool, engineering of the ECAD project, and exporting the ECAD project to PLC programming tool

• Importing ECAD project into PLC programming tool and engineering of the overall PLC project

6.2.1 ECAD engineering

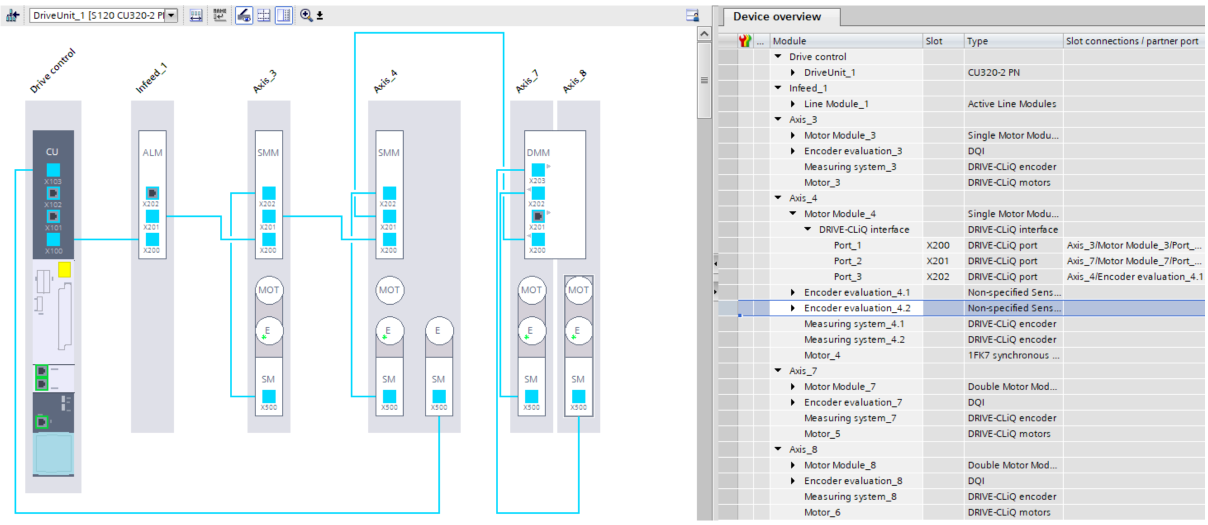
Based on the existing ECAD project, the ECAD engineer executes the complete hardware construction of the automation system including the drive configuration.



ECAD engineering of a multi-axis drive system

6.2.2 PLC engineering

From ECAD tool the hardware configuration will be imported into the PLC engineering tool.



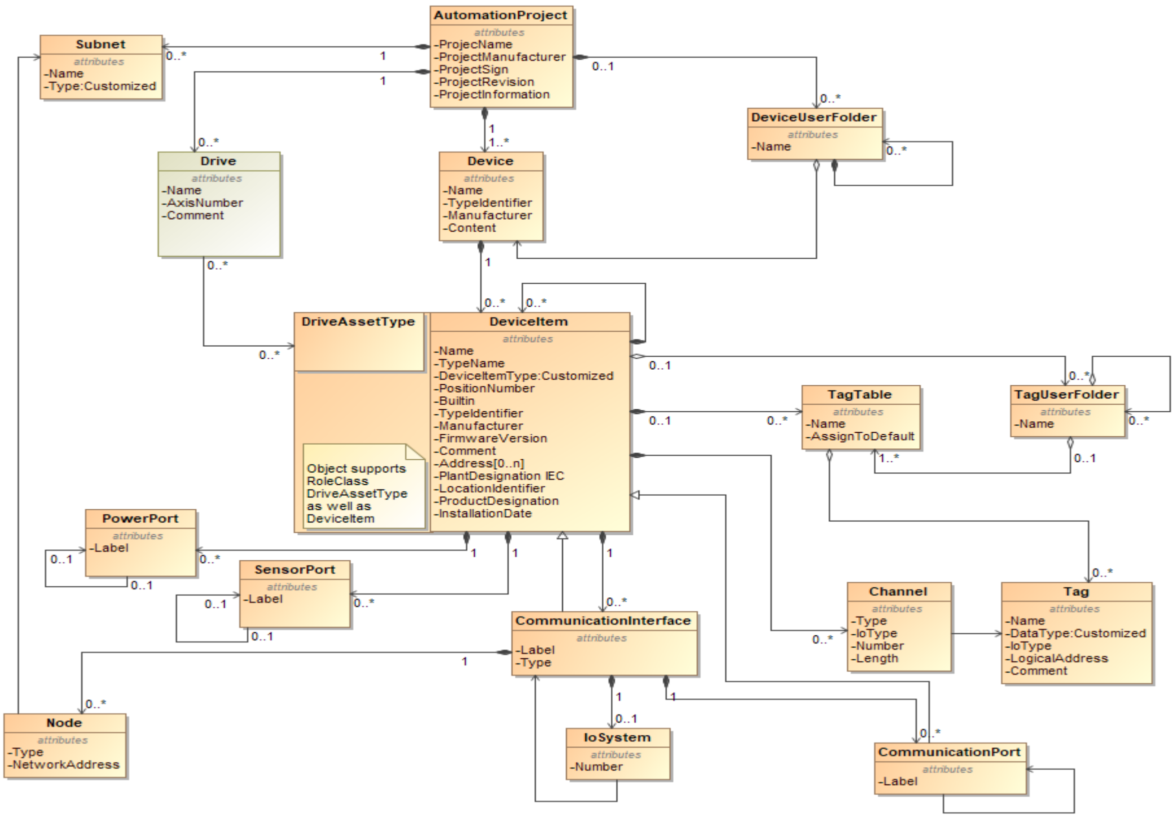
Hardware configuration of a multi-axis drive system within engineering tool

On a next step the software program will be linked to the hardware by
1. Matching the symbolic addresses of the program to the symbolic names of the hardware
2. Assigning the axis software objects used for controlling the movement of an axis to the drive's objects which act as an interface to the hardware and firmware of the drive configuration.

7. Drive Configuration data structures in AutomationML

7.1 Drive Configuration data exchange data model

The consideration of all these mentioned and already existing models leads to the following Automation Project Configuration data exchange diagram:



Objects and parameters of the Drive Configuration data exchange and their relation to the elements of Automation Project Configuration

7.1.1 Drive

A Drive object holds references to the DriveAssetTypes, which represents the electrical components of the drive. The overall dynamic behaviour of the drive configuration may be controlled by a software object, which is identified by the name of the drive.

For a PLC program, this object may also act as an interface object to the drive functionality and cyclic

I/O. The standard parameters of a Drive object are listed below.

• **Name (string)** The name of the Drive shall be unique within the automation project.

• **AxisNumber (int)** Optional information, which identifies the axis the drive object is related to..

• **Comment (string):** An optional comment for the device, using a multilanguage string The Drive objects are aggregated directly under the AutomationProject.

7.1.2 DriveAssetType

A DriveAssetType is an object to extend the definition of a DeviceItem with drive-specific information of the corresponding DeviceItem. It is also used to include the interface, which i used to reference the assigned Drive object.
In later Steps the „DriveAssetType“ could be used to specify the asset type more precisely, which is provided by the DeviceItem.

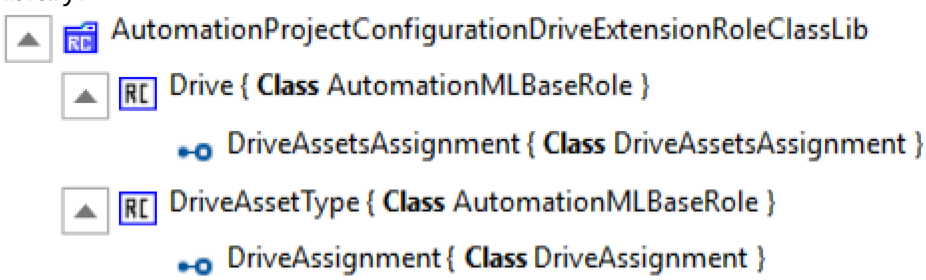
7.1.3 DriveAssignment

The interface class "Drive Assignment" is used to define the assignment of DriveAssetType to a Drive object. By using an internal link between the "Drive Assignment" interfaces of the Drive object and the DriveAssetTypes, the components are defined, which provide a dedicated functionality for the Drive.

8. Modelling of Drive Configuration data with AutomationML

8.1 RoleClassLibrary

Basement of the modelling are the required role classes. Facing the required model elements there are role classes especially required for Drive Configuration data modelling derived from role classes used for automation system modelling defined in Application Recommendation “Automation Project Configuration” The following figures represent the defined role class library:



```xml
<RoleClassLib
 Name="AutomationProjectConfigurationDriveExtensionRoleClassLib"
 ChangeMode="state">
 <Version
  ChangeMode="state">1.2.0</Version>
 <RoleClass
  Name="Drive"
  RefBaseClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole"
  ChangeMode="state">
  <Attribute
   Name="Name"
   AttributeDataType="xs:string"
   ChangeMode="state"/>
  <Attribute
   Name="Comment"
   AttributeDataType="xs:string"
   ChangeMode="state"/>
  <Attribute
   Name="AxisNumber"
   AttributeDataType="xs:int"
   ChangeMode="state"/>
  <ExternalInterface
   Name="DriveAssetsAssignment"
   RefBaseClassPath="AutomationProjectConfigurationDriveExtensionInterfaceClassLib/DriveAssetsAssignment"
   ID="6832f132-d944-4abc-b2ce-a802e9aa8578"
   ChangeMode="state"/>
 </RoleClass>
 <RoleClass
  Name="DriveAssetType"
  RefBaseClassPath="AutomationMLBaseRoleClassLib/AutomationMLBaseRole"
  ChangeMode="state">
  <ExternalInterface
   Name="DriveAssignment"
   RefBaseClassPath="AutomationProjectConfigurationDriveExtensionInterfaceClassLib/DriveAssignment"
   ID="2f157da3-224d-471c-86ec-cc98cfb1d7d6"
   ChangeMode="state"/>
 </RoleClass>
</RoleClassLib>
```


8.1.1 Drive

A “Drive” is derived from ““AutomationMLBaseRole” according to AutomationML Whitepaper -Architecture and general requirements. It is defined as follows.

Definition Drive

<table>
<tr><td>Role class name</td><td colspan="2">Drive</td></tr>
<tr><td>Description</td><td colspan="2">The role class “Drive” shall be used in order to support the structure of a Drive configuration within a project.</td></tr>
<tr><td>Parent Class</td><td colspan="2">AutomationMLBaseRoleClassLib/AutomationMLBaseRole</td></tr>
<tr><td>Path for Element reference</td><td colspan="2">AutomationProjectConfigurationDriveExtensionRoleClassLib/Drive</td></tr>
<tr><td rowspan="3">Attributes</td><td>“Name”
(AttributeDataType=”xs:string”)</td><td>The attribute “Name” defines the name of the Drive.
This attribute is mandatory.
<u>Note:</u> This attribute is modelled by the standard attribute Name of the relevant CAEX object.
<u>Note:</u> This attribute shall be unique within the automation project and aligned to the name of the axis within the PLC program. For drives referencing motors the name should be aligned to the name of the motor DeviceItem</td></tr>
<tr><td>“Comment”
(AttributeDataType=”xs:string”)</td><td>The attribute “Comment” defines a comment for the drive. The attribute “Comment” follows the Best Practice Recommendation Multilingual expressions in AutomationML.
This attribute is optional.</td></tr>
<tr><td>“AxisNumber”
(AttributeDataType=”xs:int”)</td><td>The attribute “AxisNumber” defines the number of the axis the Drive is assigned to.
This attribute is optional.</td></tr>
<tr><td>Interfaces</td><td>“DriveAssetsAssignment”
(RefBaseClassePath=”
AutomationProjectConfigurationDriveExtensionInterfaceClassLib/DriveAssignment)</td><td>This interface is used to link the Drive to the DriveAssetTypes, which represent the physical components of the drive. Exactly one DriveAssignment is allowed.</td></tr>
</table>

8.1.2 DriveAssetType

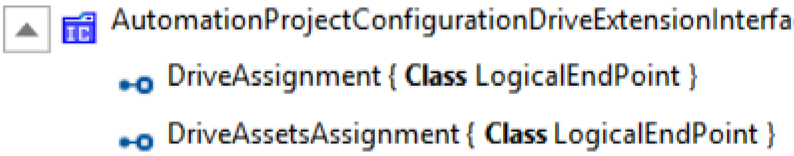
A “DriveAssetType” is derived from “AutomationMLBaseRole”. It is defined as follows.

– Definition DriveAssetType

Role class name	DriveAssetType	
Description	The role class “DriveAssetType” shall be used in order to extend a DeviceItem to hold the interface for its assignment to a Drive object.	
Parent Class	AutomationMLBaseRoleClassLib/AutomationMLBaseRole	
Path for Element reference	AutomationProjectConfigurationDriveExtensionRoleClassLib/DriveAssetType	
Attributes	None	No attributes in V 1.0.0
Interfaces	“DriveAssignment” (RefBaseClassePath=” AutomationProjectConfigurationDriveExtensionInterfaceClassLib/DriveAssignment)	This interface is used to link the Drive to the DeviceItem, which represent the physical components of the drive. Several DriveAssignments are possible. The direction of the link is not relevant. In case of connection to more than one DriveAssignments the one and only DriveAssignment shall contain all connections.

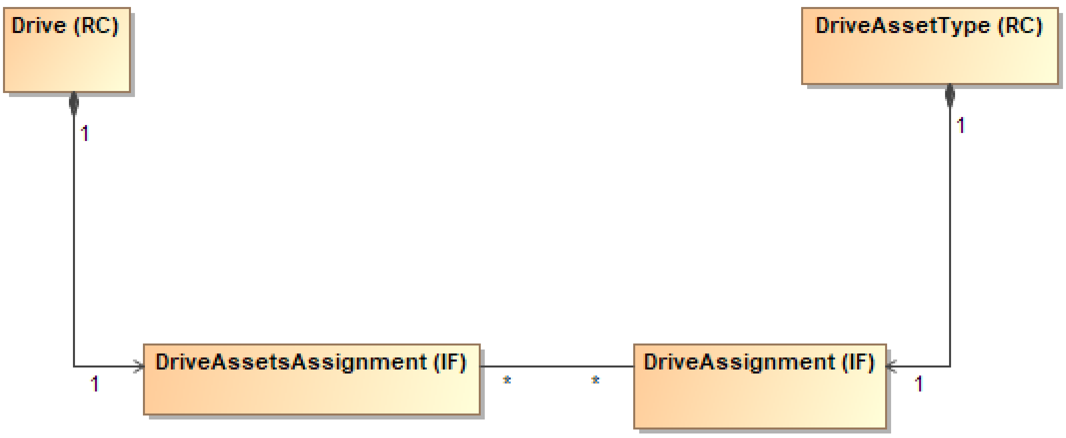
8.2 InterfaceClassLibrary

The next main base of modelling are interface classes. Facing the required model elements there are interface classes especially required for Drive data modelling derived from interface classes used from CommunicationRoleClassLib V 1.0.1 as defined in AutomationML Whitepaper – Communication. The following figures represent the interface class library.



```
<InterfaceClassLib
  Name="AutomationProjectConfigurationDriveExtensionInterfaceClassLib"
  ChangeMode="state">
  <Version
    ChangeMode="state">1.2.0&lt;/Version&gt;
  <InterfaceClass
    Name="DriveAssignment"
    RefBaseClassPath="CommunicationInterfaceClassLib/LogicalEndPoint"
    ChangeMode="state"/>
  <InterfaceClass
    Name="DriveAssetsAssignment"
    RefBaseClassPath="CommunicationInterfaceClassLib/LogicalEndPoint"
    ID="cfb4ea0b-dab8-4116-86b3-4f4fb0b1b148"
    ChangeMode="state"/>
&lt;/InterfaceClassLib&gt;
```

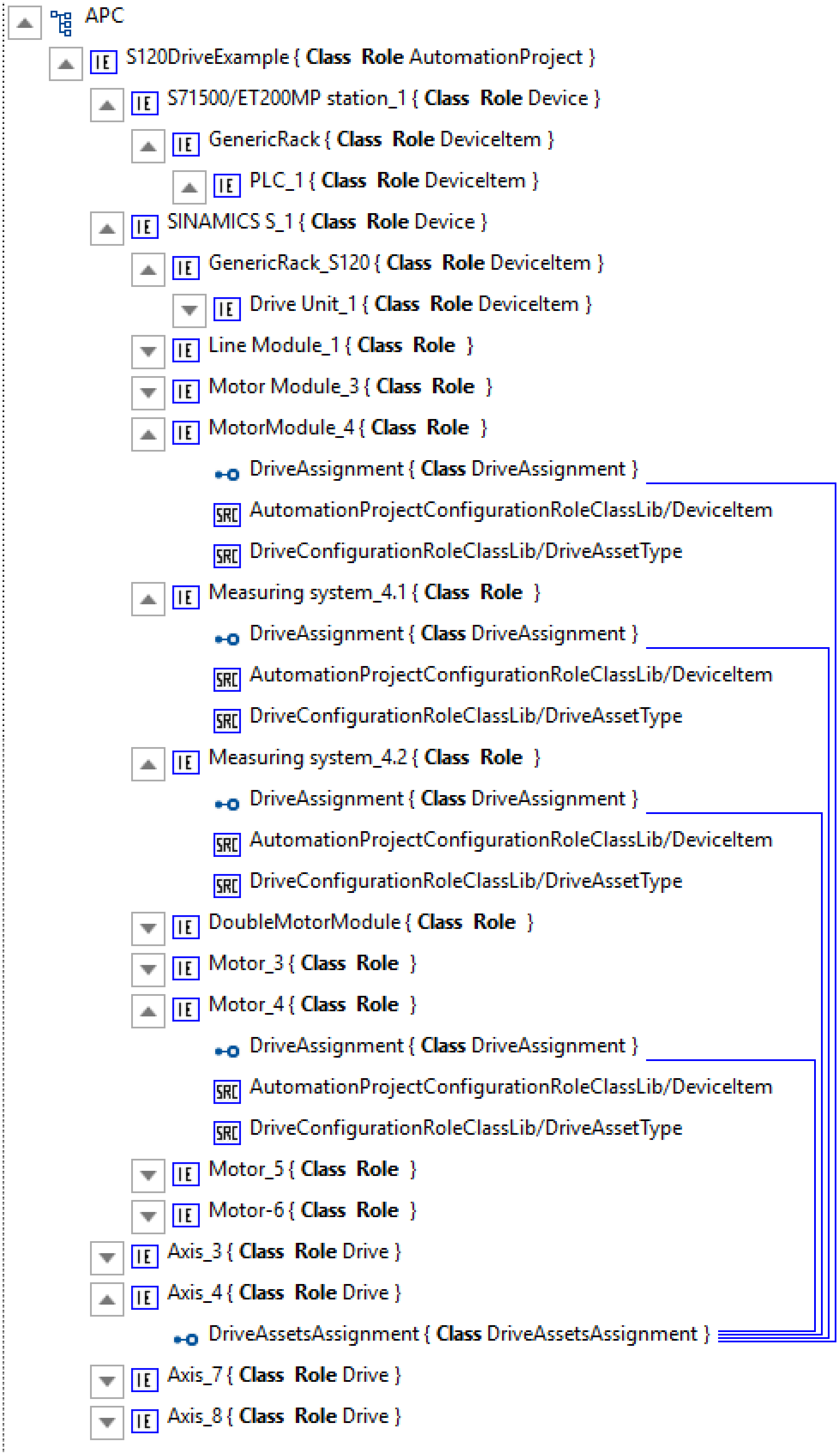
Each Drive object has exactly one external interface of the class DriveAssetsAssignment which has internal links to all of the associated DriveAssetTypes. A DriveAssetType has exactly one external interface of the class DriveAssignment which has internal links to the Drive object that it is associated with.



9. Appendix A Appendix A: Automation-ML

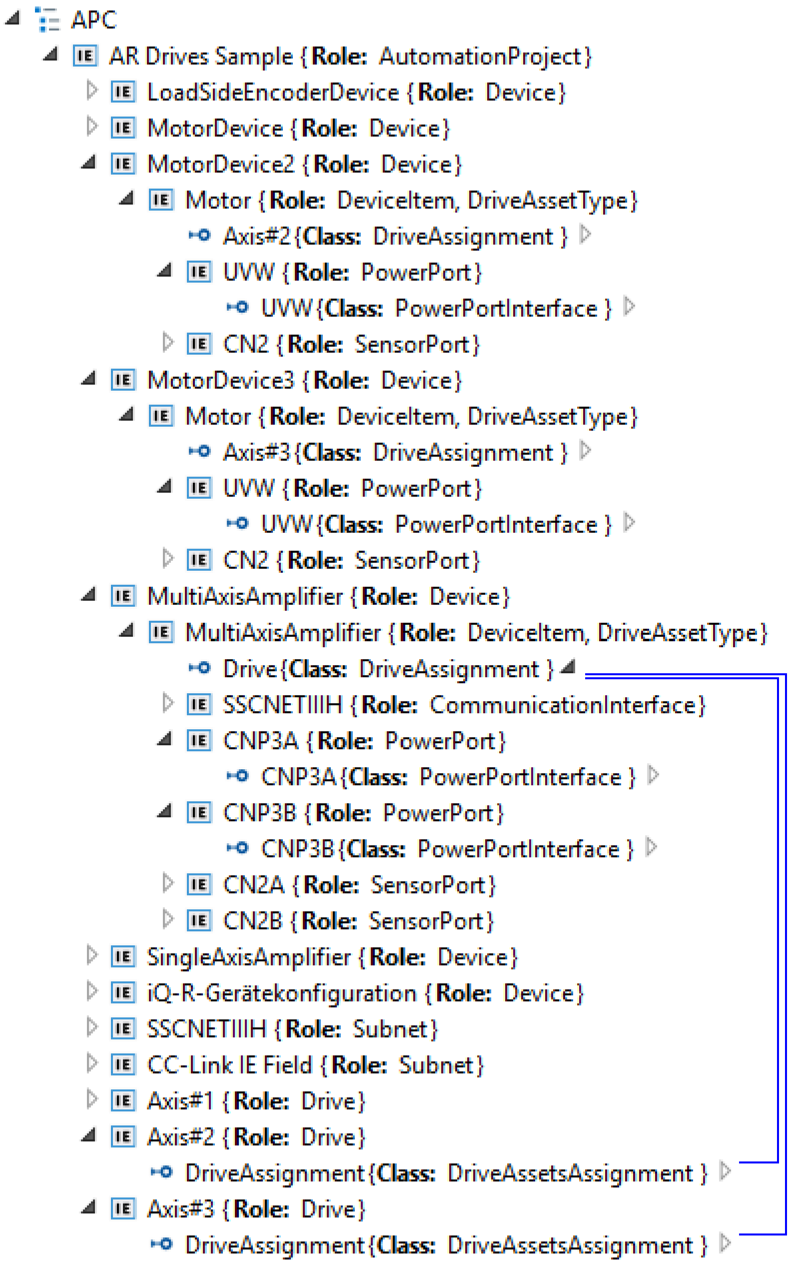
9.1 Drive Configuration with infeed, inverter and sensor modules (PLC with S120)

This part shows a modular multi-axis drive with 4 drive objects consisting of control unit, infeed, inverters and motors with attached measurement systems. The different Drive objects are linked to the device items which represent the electrical parts of the drive.


APC
S120DriveExample { Class Role AutomationProject }
S71500/ET200MP station_1 { Class Role Device }
GenericRack { Class Role DeviceItem }
PLC_1 { Class Role DeviceItem }
SINAMICS S_1 { Class Role Device }
GenericRack_S120 { Class Role DeviceItem }
Drive Unit_1 { Class Role DeviceItem }
Line Module_1 { Class Role }
Motor Module_3 { Class Role }
MotorModule_4 { Class Role }
DriveAssignment { Class DriveAssignment }
AutomationProjectConfigurationRoleClassLib/DeviceItem
DriveConfigurationRoleClassLib/DriveAssetType
Measuring system_4.1 { Class Role }
DriveAssignment { Class DriveAssignment }
AutomationProjectConfigurationRoleClassLib/DeviceItem
DriveConfigurationRoleClassLib/DriveAssetType
Measuring system_4.2 { Class Role }
DriveAssignment { Class DriveAssignment }
AutomationProjectConfigurationRoleClassLib/DeviceItem
DriveConfigurationRoleClassLib/DriveAssetType
DoubleMotorModule { Class Role }
Motor_3 { Class Role }
Motor_4 { Class Role }
DriveAssignment { Class DriveAssignment }
AutomationProjectConfigurationRoleClassLib/DeviceItem
DriveConfigurationRoleClassLib/DriveAssetType
Motor_5 { Class Role }
Motor-6 { Class Role }
Axis_3 { Class Role Drive }
Axis_4 { Class Role Drive }
DriveAssetsAssignment { Class DriveAssetsAssignment }
Axis_7 { Class Role Drive }
Axis_8 { Class Role Drive }

9.2 Drive Configuration with double multi-motor-module

This example shows a drive configuration including a multi-axis servo amplifier that controls two motors. The multi-axis amplifier is linked to the corresponding Drive objects using the DriveAssignments interfaces and internal links**.**



finally we can run aml using **''twinning ClassLibs_V1_2_0.aml''** that you can find this file with folder of the project

10. References

[1] https://www.automationml.org/o.red.c/dateien.html

[2] R. Drath, A. Luder, J. Peschke, and L. Hundt.
Automationml - the glue for seamless automation engineering.
In 2008 IEEE International Conference on Emerging Technologies and Factory Automation, pages 616–623, Sept 2008.

[3] https://dl.acm.org/doi/10.1145/3198458.3198464

[4]https://www.researchgate.net/publication/319888412_A_Review_of_the_Roles_of_Digital_Twin_in_CPS-based_Production_Systems

[5]https://www.t-systems.com/en/best-practice/03-2018/focus/ethical-issues/use-cases/digital-twin-840488

[6]https://www.i-scoop.eu/digital-twin-technology-benefits-usage-predictions/

[7]https://link.springer.com/chapter/10.1007/978-3-030-25312-7_14