# Data Structure for Virtual Commissioning - Exchange Relevant Information Focusing on Integration of Modular Component Models and Wide Compatibility

Dominique Geiger, Benjamin Massow (supervisor), and Thomas Hausberger (supervisor)

***Abstract*—In this paper a concept for component libraries is created consisting of CAD data, simulation models, example control code and documentation.**

**Virtual commissioning is becoming increasingly important in the engineering process of plants and machines. This is the result of time savings and thus reduced costs in product development. However, implementation can easily become complex.**

**The behaviour modeling of the different components can be done in several different tools, mostly depending on the manufacturer. The CAD data can also be available in different file formats and are thus no exception of this rule. This variety of possibilities increases the complexity of the integration process enormously. However, in order for virtual commissioning to be efficient, the integration of the used components should be as simple as possible.**

**Aiming to solve this problem, an exemplary library structure is developed in this thesis, which consists of the behaviour model, CAD data and code snippets with related documentation. The basis for this are freely available and common interfaces. This ensures that the widest possible range of applications is achieved. The practical use of this library is further demonstrated in a practical application.**

**In the example shown, the behaviour model and control are executed on two different machines ensuring real-time capability and forming a *hardware in the loop* (HIL) system. The behaviour model is generated in *Simulink* and integrated into the PLC run-time.**

**As shown, using the developed library, the process of virtual commissioning can be simplified without the need for additional expertise. However, there is still room for improvement especially in the exchange of the CAD data with a kinematization and the integration of behaviour models in a PLC.**

***Index Terms*—Virtual Commissioning, PLC, Automation, Modularity, Data Library.**

D. Geiger studies at MCI Innsbruck, e-mail: dm.geiger@mci4me.at.

## I. Introduction

VIRTUAL commissioning is gaining relevance in many industry branches due to its advantage of reducing time and therefore costs. However, the creation and especially the integration of plant models is a hurdle. A unified data structure consisting of the geometry, behaviour model and exemplary PLC code can reduce this issue.

Aim of this paper is to develop a data structure consisting of kinematized CAD assemblies, physical behaviour models and PLC code for an easier integration in a virtual commissioning. With this data structure, the exchange of components between supplier and customer can be standardized and, in addition, the setup of a virtual commissioning can be simplified.

## II. State of the Art

### A. Examples for a Virtual Commissioning

A virtual commissioning can be done in different ways, whereby the chosen method also strongly depends on the specific application. In this chapter, various possibilities will now be briefly explained.

*1) Visualization in the PLC Development Environment:* The *human-machine interface* (HMI) offers a quick possibility for testing the control software. In most cases, a *graphical user interface* (GUI) can be created directly in the development environment in which current values and outputs of the control software can be displayed and inputs can be set. In this approach, the human developer imitates the behaviour of the real plant and therefore tries to identify potential problems in the software. Due to the human interaction, fast processes and reactions can only be tested to a limited extent. However, basic code sections and functions can be evaluated during development and in the process of a virtual commissioning.

*2) Digital Twin using an Physics Engine:* The next step is the use of an external physics engine such as *Unity*. Especially by using existing libraries and well documented interfaces a simple model can be created in a short time. In addition, the handling of the model can be simplified by using the original geometry data. A disadvantage of this method compared to the direct visualization in the PLC development environment is the requirement of an additional communication between the PLC run-time and the physics engine. Nevertheless, depending on the implementation, this procedure can also be carried out in real-time. By using the original CAD data, a visually appealing model can be created and controlled via the PLC. This is an important advantage especially in presentations with customers or in sales, in order to be able to address people from the non-technical area as well. An example for the implementation in *Unity* and *TwinCAT* is [1]. Here, a digital twin is tested and used in a HIL system, achieving a communication time of $10\,\mathrm{ms}$ and a step time of the simulation in *Unity* of $5\,\mu\mathrm{s}$.

*3) Digital Twin in a Simulation Software:* The use of simulation software such as *Matlab/Simulink* is particularly advantageous for complex systems or in control engineering. Complex systems can be easily created using mathematics and a GUI. Furthermore, it improves the overview and minimizes the time needed to maintain the models. In this method, the model is executed in the simulation software and for this reason also requires additional communication to the PLC run-time. If the system is supposed to be real-time capable, special hardware or precautions are often required.

### *B. Required Information for Setup*

For the identification of the required information for a virtual commissioning, the general process of product development has to be examined. The development of a product includes several stages in different engineering disciplines. These stages can partly be developed in parallel to save time and costs in the development process. However, special attention must be paid to the dependencies between the stages. The first phase of product development includes the general boundary conditions like defining objectives, constraints and interfaces. The next step is to develop the product's hardware usually consisting of mechanical and electrical components. The hardware must be developed in collaboration between both disciplines, since changes in the mechanics, for example, often also lead to changes in the electrics. Parallel to the development of the hardware, the process of software development can be started already. For the completion of the software, however, the hardware must already have been finalized. The final step in product development is an optional virtual commissioning followed by actual commissioning.

Information from all phases of product development are required for the successful execution of a virtual or a real commissioning:

*1) Process Design:* Interfaces to other parts of the plant, but also time limits and throughput quantities. The selection and characteristics of used sensors and actuators also belong to this area. The characteristics of the used hardware is particularly important in the creation of behaviour models, where the information is given in data sheets or already integrated in models.

*2) Mechanical Engineering:* The general structure of the mechanics and the kinematics of the individual components are defined in this step. The design of the product and the used materials determine the physical behaviour and result in multiple CAD files.

*3) Electrical Engineering:* The configuration of the PLC with the used terminals is part of this section. Especially interesting for the development of the software is the mapping between the inputs and outputs of the terminals with the connected devices. The documentation of the mapping and the structure of the PLC can be shown in a tabular form or in a dedicated object in the development environment of the PLC.

*4) PLC Coding:* If more complex components are used in the plant, often these are addressed via their own interface. The documentation and the knowledge of handling these interfaces is important for the creation of the software but also for the later commissioning. In the best case examples exist, which explain the handling and therefore ensure a faster implementation. But also an example PLC code of simpler components can often be advantageous.

### C. Criteria for a Data Structure

Based on these fundamentals of a virtual commissioning, criteria for a data structure can be identified. With the help of these criteria, the efficiency of a data structure can be determined, whereby these should be fulfilled as good as possible. The identified criteria are:

- Independent of used hardware setup (HIL/SIL).
- Represent relevant information.
- Modular layout in order to represent components of a real plants.
- High compatibility with common tools.
- Low level of additional knowledge required.

## III. Proposed Data Structure

Based on the previous findings, a data structure is developed and presented in this chapter. The main objective of this data structure is a high compatibility and a modular design in order to be able to represent the individual sub-components of a real plant. The data structure itself should contain all necessary information required for a virtual commissioning.

### A. Layout of the Data Structure

The actual layout of the data structure is kept simple and consists of a root folder with several directories for the relevant information as shown in Figure 1. The sub-directories contain the information of kinematization and CAD, physical model, PLC source code and documentation. An optional *ReadMe* file used for general information and revision control of the structure itself. Finally the root folder is compressed into a *.zip* file for an easier distribution. Due to the plain and simple structure, it can be easily extended and modular exchanged.

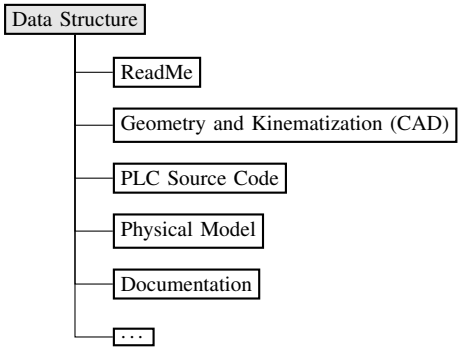


Fig. 1: Layout of proposed data structure containing required information for a virtual commissioning.

### B. Selected File Formats

With the layout of the data structure appropriate data formats for representation of CAD assemblies, physical models and PLC code are now selected. For this purpose different possibilities of implementation are tested on the basis of the used software shown in Table I.

TABLE I: Used software for selection of suitable data formats for the data structure.

| Area | Used Software |
|---|---|
| CAD | Autodesk Inventor Professional 2022<br>Autodesk 3ds Max 2022 |
| PLC | TwinCAT 3 Version 3.1.4024.25 |
| Modelling | Matlab R2020b, Simulink |

*1) CAD Data:* The exchange of CAD data between different tools is usually done via neutral formats such as *.step* or *.iges*. However, only the pure geometry is supported and transferred via these formats and an information loss of the kinematization occurs. In many areas and also in virtual commissioning, kinematization and thus its exchange is, however, an important prerequisite. The industry has recognized this problem of the missing interface of the kinematization and is therefore working on different solutions.
For example, existing CAD data formats can be extended to support features and kinematization. For this purpose, this paper [2] proposes a possible extension of the *.step* format in order to include the kinematization. A second possibility is the development of a new and neutral data format. In order to be able to use this format, an integration into existing CAD tools can be done or, alternatively, a conversion of the native formats with the help of translators is also an option. However, both methods are feasible but require a major effort in technical and organizational terms. This paper [3] shows the conversion of native features in the design of a part into a neutral format and back into a second CAD software. The same approach could be used to export the kinematics as well as the features, as shown in this paper [4]. As an alternative, the kinematics could be saved in an additional file and linked to the original CAD data as shown for example in this paper [5]. A promising solution is the *COLLADA* format, which supports kinematization starting from version 1.5 [6]. The *COLLADA* format, released back in 2004, is based on *.xml* documents and is mainly used in the entertainment and gaming industry suffering from the same problem with a large number of incompatible tools. The use in the manufacturing industry is currently not attractive, because suitable tools for the conversion to and from native formats are still missing.
Since there is no established solution for exporting kinematic CAD assemblies, there are two options for the data structure: Either a neutral format such as *.step* is used, which has a high compatibility but does not support kinematics, or a native format is used, in which compatibility is reduced but kinematization is preserved.

*2) Behaviour Model:* Similar to the geometry data, there is a variety of possible file formats for the description of a physical behaviour model. This is mostly dependent on the discipline and the simulation software used. However, especially in the field of co-simulation a universal interface is required to simplify data exchange due to the combination of multiple engineering disciplines. For this reason, the *Functional Mockup Interface* (FMI) was defined by *Modelica Association* already in 2010 and is currently in fact the default format for exchanging models. Currently this interface is available in version 3 and is already supported by more than 170 different tools [7]. For this reason, the *FMI* format is selected for the data structure to represent physical models in *.fmu* files.

*3) PLC Code:* In the area of control engineering and automation using PLCs, the basis is mainly the international IEC 61131 standard. This standard is based to a large extent on the organization *PLCopen*, which has set itself the goal of increasing the efficiency in the creation of control software and to be platform-independent between different development environments. To make this possible the PLC project with its code and libraries are saved as *.xml* files and therefore can be exchanged without problems. This exchange format was standardized in 2019 in Part 10 of the IEC 61131 standard. Thanks to the

international validity, many manufacturers rely on this standardization and offer converters to and from the *PLCopen* format.

*4) Overview of Selected Data Formats:* Based on these findings an overview of the selected data formats is found in Table II.

TABLE II: Chosen file formats for the data structure.

| Information | File Format |
|---|---|
| • CAD as pure geometry | Neutral like *.step* |
| • CAD with kinematization | Native like *.iam* or *COLLADA* |
| • Physical model | FMI as *.fmu* |
| • PLC code | *PLCopen* as *.xml* |

### C. General Workflow of Using the Data Structure

The workflow when using the proposed data structure is shown in Figure 2 and consists of two main parts: Generating the structure (export) and using the structure in a virtual commissioning (import).

The export of the data into the structure is more straightforward and thus easier to accomplish compared to the import. The data structure can usually be generated in just a few steps, with CAD data being the exception to this general rule. In principle, the greatest effort is required when using the data structure in the field of CAD data, due to the lack of an interface for saving the kinematics of an assembly. If only the geometry should or can be transferred, a neutral format like *.step* or *.iges* is recommended, which are supported by most of the CAD tools. Until the *COLLADA* format or alternatives are established in industry, exporting kinematics is done easiest in native CAD formats. In comparison, the export of the behaviour model, the control code of the PLC and, if necessary, a technical documentation do not cause major problems and are supported by the majority of available software.

The process of importing the data structure, in contrast to exporting, is more complex, where the individual steps strongly depend on the used target system. In the case of CAD data, it is necessary to distinguish between three possible approaches: the assembly is not kinematized (for example *.step*), the assembly is kinematized in neutral format (for example *COLLADA*) and the assembly is kinematized in a native format (for example *.iam*). The native format provides the least effort for integration, followed by plain geometry without kinematization (depending on the complexity of the assembly). The greatest effort represents the import of a *COLLADA* file, due to the lack of conversion tools. The import of the PLC code is done in the majority of PLC systems without any difficulties thanks to the international standardization of the programming language. As a result sample code can be integrated into an existing project fast and without any difficulties. However, with the behaviour model, a dependency to the used software is once again recognizable. In the best case, the *.fmu* object can already be executed directly in the PLC and linked to the inputs and outputs of the hardware. Alternatively, the model can be executed in a simulation software, which again communicates with the PLC run-time.

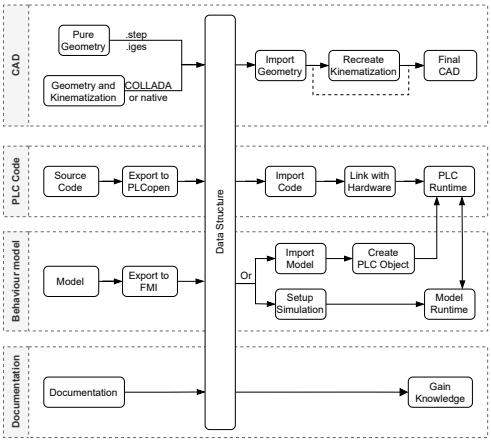


Fig. 2: General workflow of proposed data structure consisting of the most important aspects for the export and import of the data structure.

## IV. Exemplary Application

The practical use and individual steps in the implementation of the proposed data structure are shown by means of an example using the same software as shown in Table I.

### A. Considered Plant

In order to represent a real plant a modular *Teaching Factory* is used as shown in Figure 3. The aim of this plant is to give an education platform in order to get familiar with automation technology with the help of a PLC and its programming.

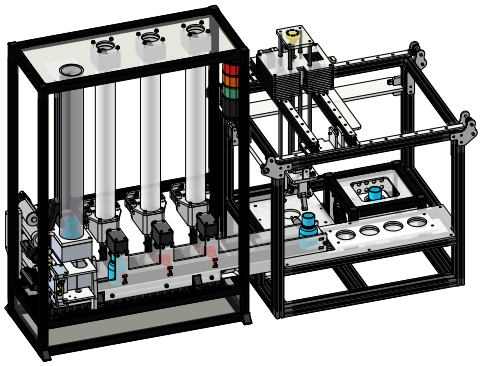

Fig. 3: The used *Teaching Factory* representing a filling station. On the left is the module *Separation* followed by the *Conveyor Belt* with three *Dosing Units*. The right half contains the *Cartesian Gripper* with a *Load Cell* (left), *Thermal Processing* (rear right) and an output storage (front).

### B. Generate the Data Structure

The first step is the generation of the proposed data structure consisting of CAD data, behaviour model, PLC source code and optional documentation. This data export is done from the view of a supplier for each module of the plant.
For exporting the CAD data, this example uses the native format of *Inventor* for kinematized models. That is based on the assumption that the transmitter and receiver of the data structure both use *Inventor*. If the assembly is purely static and thus no kinematicization is required, the *.step* format is used for exchange.
In addition to the *.fmu* files, the behaviour models are included also as original *Simulink* files into the data structure. This is done because the required product *TE1420* to integrate the *.fmu* models into *TwinCAT* is still under development [8]. For this reason, the models are integrated into *TwinCAT* via code generation from *Simulink* [9]. The intended product, as well as the alternative generate a *TcCom* object, which is then linked to the inputs and outputs of the PLC. For that reason, the result of the two methods is identical, but the source format is different. The interface of the models reproduces the signals of the real hardware and describes the given system. The level of detail of the models can vary depending on the requirements of the simulation.
The source code of the PLC is exported to the *PLCopen* format for exchange.

The collected information is finally bundled in the data structure as shown for an exemplary module in Figure 4.

### C. Import the Data Structure

The integration of the data structure in a virtual commissioning is also done for the areas: CAD assembly, behaviour model and PLC code.
In the case of CAD data, the integration is a standard process for an average user of *Inventor*. If the kinematization is lost during the export, for example when only the pure geometry is exchanged, it has to be recreated by the customer. The integration of PLC code into *TwinCAT* is similarly simple due to the standardization of the *PLCopen* format. As already mentioned, the product for integrating the behaviour models is under development and therefore code generation is used as an alternative. For this purpose, the original model from *Simulink* is used and converted into a *TcCom* object in *TwinCAT* [10]. In this way, a separate object is created on the PLC

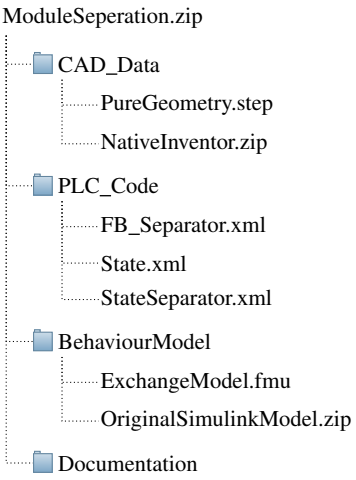


Fig. 4: Data structure of module *Separation* consisting of CAD data as pure geometry and including kinematization in a native format, PLC code and the behaviour model as *.fmu* and original *Simulink* files. Other documentation is in this case not required.

for each module, which is finally connected to the inputs and outputs of the control code.
With the integrated models and PLC code, a virtual commissioning can now be performed. In this example the model of the plant and the control code are executed on two different machines therefore forming a HIL system with a real-time communication between them. Using the data structure in a *software in the loop* (SIL) system is also possible.

## V. Results and Evaluation

The result of this paper is the proposed data structure bundling information for a virtual commissioning. The structure is modular designed and data is stored in compatible formats. The practical use of this data structure is evaluated by means of a example shown in section IV using the criteria from subsection II-C.

Through the evaluation it is stated that the data structure can basically be used independent of the hardware setup for example in a HIL system. The data required for virtual commissioning, such as CAD assemblies, behaviour models, PLC code and documentation, can be represented in the data structure without problems. As shown in the example, individual modules of a plant can be reproduced with the data structure and finally integrated into a higher-level system. However, there is still room for development in the compatibility of the data formats, especially in the area of kinematized CAD data and in the integration of *.fmu* models in a PLC project. While the integration of behaviour models in the case of *Beckhoff* is only a matter of time, the exchange of kinematization in CAD data represents a bigger challenge. In this case, a practicable solution has not yet been established in the industry. In terms of the requirements for the additional know-how needed, no further skills are required in the case of the CAD data and the pure PLC code. When creating the behaviour models, basic knowledge of programming a PLC is an advantage but not absolutely necessary. However, the integration of the behaviour models into the PLC project requires additional knowledge, which can be acquired in a seminar, for example. The result of this evaluation is summarized in Table III and is valid for the used software versions shown in Table I, where upward compatibility is likely but downward compatibility is not guaranteed.

TABLE III: Evaluation of the data structure with respect to the shown example from bad (○○○) to good (●●●). Although the defined information is represented in the data structure, compatibility could be further improved.

| Criterion | Evaluation |
|---|---|
| Independent of hardware setup (HIL/SIL) | ●●● |
| Represent relevant information | ●●● |
| Modular layout to represent components | ●●● |
| High compatibility | ●○○ |
| Low level of additional knowledge | ●●○ |

## VI. Summary

In the scope of this master thesis, a data structure for the exchange of relevant information for a virtual commissioning is developed. This structure is focused on high compatibility between different tools and has a modular structure. The covered information of the data structure includes the mechanical design represented by the CAD data, physical behaviour models of different used components and source code for the control via a PLC. Additional information such as data sheets can be added to the data structure as required.

The presented data structure is evaluated on the basis of an example for its usability. The result of this evaluation shows potential for improvement, especially in the exchange of kinematized CAD assemblies. Besides the native formats there is no established solution available at the moment. Furthermore, the integration of *.fmu* models into a PLC run-time is also under development in the case of *Beckhoff* but suitable alternatives are found.

All in all the data structure works in general and shows its usability in the example. As a result of using the data structure the difficulties in setting up a virtual commissioning can be reduced with the advantage of minimizing time and costs.

## VII. Outlook

The next possible steps of this development are on the one hand further research on the exchange of kinematized CAD assemblies based on a neutral data format and on the other hand further evaluation of the data structure with respect to bigger use cases and additional software for the integration of the behaviour models in the PLC. This can also further investigate compatibility of different versions of the software used.

## References


[1] H. Sangvik, "Digital Twin of 3d Motion Compensated Gangway Use of Unity Game Engine and TwinCAT PLC Control for Hardware-in-the-Loop Simulation," Master's thesis, University of Agder, 2021.

[2] "Standardized data exchange of CAD models with design intent," *Computer-Aided Design*, vol. 40, no. 7, pp. 760–777, 2008, current State and Future of Product Data Technologies (PDT).

[3] B. Kim and S. Han, "Integration of history-based parametric translators using the automation APIs," *International Journal of Product Lifecycle Management*, vol. 2, no. 1, p. 18, 2007.

[4] Y. Kim, H. Lee, M. Safdar, T. A. Jauhar, and S. Han, "Exchange of parametric assembly models based on neutral assembly constraints," *Concurrent Engineering*, vol. 27, no. 4, pp. 285–294, Aug. 2019.

[5] D. Wang, K. Cao, and J. Wang, "Study on analyzing and remodeling assembly constraints of CAD models from the heterogeneous system," in *International Conference on Computer Vision, Application, and Design (CVAD 2021)*, Z. Zhang, Ed. SPIE, Dec. 2021.

[6] M. Barnes and E. L. Finch, "COLLADA - Digital Asset Schema Release 1.5.0 Specification." Sony Computer Entertainment Inc., 04 2008.

[7] Modelica Association Project, *Functional Mock-up Interface Specification, Version 3.0, 2022-05-10*.

[8] Beckhoff Automation GmbH, *TE1420 — TwinCAT 3 Target for FMI*, product data sheet.

[9] ——, *TE1400 — TwinCAT 3 Target for Simulink*, product data sheet.

[10] Beckhoff Information System. TE1400—TwinCAT 3 Target for Simulink. Visited on 2022-05-26. [Online]. Available: https://infosys.beckhoff.com/content/1033/te1400_tc3_target_matlab/index.html?id=7328785815492855617



**Dominique Geiger** is Master student in the Department of Mechatronics at MCI Innsbruck/Austria.