3-4-2015

Function Based Engineering with AutomationML - Towards better standardization and seamless process integration in plant engineering

Florian Himmler

Function Based Engineering with AutomationML –
Towards better standardization and seamless process integration in plant engineering

Florian Himmler[1]

[1] evosoft GmbH & University of Erlangen-Nuremberg, Nuremberg, Germany
florian.himmler@{evosoft.com, fau.de}

Abstract. Due to shorter product lifecycles in combination with increasing product complexity, more and more enterprises are considering using the *Digital Factory*. One important part of the *Digital Factory* is the *System Layout/System Design* process, which includes the engineering of industrial plants – a very complex and cost-intensive domain. In this design science oriented paper, we developed a concept for standardized application interfaces in plant engineering by using AutomationML in combination with the function-based standardization framework. This concept enables companies to foster the standardization and modularization of industrial plants as well as the seamless integration of all IT systems involved in the engineering process. We validated our developed methodology and its artifacts by applying it in two real business settings, and found that overall it has a huge positive impact on *process cycle times*, *data redundancy*, *data quality* and *extensibility*.

Keywords: plant engineering, AutomationML, standardization, modularization, function-based engineering, integrated engineering, data integration

1 Introduction

Today, industrial enterprises are facing shorter and shorter product lifecycles in conjunction with increasing complexity of the products that they manufacture. These changing basic conditions require enterprises to enhance their planning efficiency as well as their planning quality – especially with regards to the engineering of their producing plants. One approach to reach these goals is to introduce the *Digital Factory*. Derived from the Association of German Engineers' (VDI) definition of the term [1], the *Digital Factory* is defined as an IT system capable of digitally planning, controlling and optimizing all resources and activities related to a product which are performed beginning with its development and ending in the order processing – prior to the start of the real production of the product [2]. Based on the reference process for the *Digital Factory* published by [2], the *System Layout/System Design* process is an essential part of the concept. It covers all activities that are needed to plan the factory building, its equipment and the systems used for production and therefore also includes the plant engineering process.

12th International Conference on Wirtschaftsinformatik,
March 4-6 2015, Osnabrück, Germany

Himmler, F. (2015): Function Based Engineering with AutomationML - Towards better standardization and seamless process integration in plant engineering, in: Thomas. O.; Teuteberg, F. (Hrsg.): Proceedings der 12. Internationalen Tagung Wirtschaftsinformatik (WI 2015), Osnabrück, S. 16-30

Prior research on this topic has shown that in order to efficiently engineer plants over the long term, it is crucial to implement interdisciplinary engineering processes that are based on an integrated system landscape. However, there are currently two main factors which preclude such a scenario:

- The degree of integration of the different disciplines involved in the engineering process (e.g. mechanics, electrics and software) is insufficient (e.g. due to insular thinking, globally distributed teams, a lack of process alignment, etc.) resulting in information deficiencies and inconsistencies between them [3–5].
- There are heterogeneous system landscapes with a low degree of integration between the software tools, leading to redundant data, data inconsistencies and high development costs when changes are required [4, 6–8].

In this design science oriented paper, we will introduce an integration methodology and a software artifact (referred to as "concept" in the following) for standardized application interfaces in plant engineering based on AutomationML in combination with the function-based standardization framework for the plant engineering domain. Particularly by combining the AutomationML based integration concept with the standardization framework the following aspects can be improved:

- Foster the standardization and modularization of industrial plants throughout the complete engineering lifecycle.
- Foster the integration of all IT systems involved in the engineering process.

2 Related Work

The term *plant engineering* describes "that branch of engineering which embraces the installation, operation, maintenance, modification, modernization, and protection of physical facilities and equipment used to produce a product or provide a service" [9].

Based on the problems currently existing in plant engineering regarding the high number disciplines involved and the heterogeneous system landscapes, there are currently two main challenges for companies from the plant engineering domain:

- Development of a concept for an interdisciplinary integration of all disciplines involved in the engineering process [10, 11].
- Development of an integration concept which improves the degree of integration between the tools of existing system landscapes by using standardized application interfaces [4, 7].

To implement standard application interfaces such as these, it is necessary to base them on a standardized and *Neutral Data Exchange Format* [12]. Currently, AutomationML (AML) is the most promising approach to such a *Neutral Data Exchange Format*. AML is a neutral and XML-based data format, which is designed to store and exchange plant layout data. Its primary goal is the data exchange between heterogeneous engineering tools [13]. AML is a combination of different standard formats such as CAEX, COLLADA and PLCopen. It enables topology, geometrics, kinemat-

ics and logics information to be exchanged. This paper will exclusively focus on the exchange of topology information.

There are already various AML-based integration concepts available in the current literature. In [14] a concept for an engineering table is developed, which allows the interactive planning of a plant and the transfer of the information using AML. An approach towards the modelling of graph-based structures in AML is introduced by [15]. In [16] a methodology is defined, which is capable of describing and exchanging manufacturing processes using AML. The authors of [17] describe a concept and an initial implementation of an AML-based server architecture for computer aided production engineering. Additionally, they discuss the development and integration of a web-based AML editor for the server.

The concepts mentioned above all have the potential to solve some specific integration problems in the plant engineering domain. However, all of these concepts are specific solutions to some isolated problems. A concept, which attempts to find a solution for the interdisciplinary collaboration problems in combination with a highly integrated system landscape, has not been available up until now.

3 Integrated and Function-Based Engineering with AML

3.1 Function-Based Standardization Framework

Function-Based Engineering is the process of breaking down a plant or sub-plant into separate units from a functionality point of view. The result is a structured representation of the functions that the plant is offering [18]. The corresponding standardization framework is based on a three-step process model, which guides users through the standardization process. This process model contains the three different steps *Define*, *Standardize* and *Realize*. Please refer to Table 1 for a detailed description of the process steps.

Table 1. Function-based standardization process description

Process	Description	Result
Define	Break down the plants into its functional structures. The result is a list of functions that the plant could potentially consist of. It is crucial to combine associated functions to create reasonable functional units, which are able to be reused in multiple plants.	Generic *functional structure* of the *Plant*
Standardize	Definition of standardized (sub-) plants based on the functional structures defined in the preceding step. For each of the functional structures n-different standardized structures can be assigned. The standardized (sub-) plants are subsequently available for realization in specific projects and can be reused multiple times.	*Standardized Plant Structures*
Realize	Instantiation of the standardized (sub-) plants defined in the previous step to use them in a specific context. To be able to produce the plant, every function needs to be assigned exactly one mechatronic object that is able to fulfill the function.	*Realized Plant Structure* based on standard

To be able to realize such a function-based standardization framework, we developed an object model that is capable of storing all the information needed to engineer

a plant from a functional point of view, standardize the plant, define with which objects the functions can be realized and provide different views of a plant in order to satisfy the needs of different roles involved in the process (e.g. engineer, production planner etc.). This object model consists of six different object types (*Customer Requirement*, *Plant*, *Function*, *Mechatronic Object*, *Feature* and *Purchasable Object)* and a couple of possible relationships between these objects.

Fig. 1 shows how the engineering of plant evolves during the process steps *Define*, *Standardize* and *Realize*, and how the artifacts developed in each step are related to each other. During the *Define* phase, the object/plant is broken down into its functional structure. The result of this step is a list of all functions the plant could potentially consist of (*Generic Plant GP*). Each function can be defined as being optional. Functions marked as optional could be removed during the subsequent standardization process. The subsequent *Standardize* step standardizes the objects/plants based on the functional structure defined in the preceding step (*Standard Plant GP*). These standardized objects/plants are available for realization within the next step and can be reused in multiple projects. During the final *Realize* phase the standardized objects/plants defined in the preceding step are instantiated in order to use them in a specific project or production context.

The objects and its structures, which were defined during the three process steps *Define*, *Standardize* and *Realize*, can be divided into project independent data (master data) and project specific data (project data). All objects generated during the *Define* and *Standardize* phase are considered to be master data. This means that the data is only defined once, and can be reused multiple times during the following step without requiring any additional engineering efforts. The objects generated during the *Realize* phase are considered to be project-specific data (project data). This means that for each of the plants/objects engineered during this step, an individual object needs to be instantiated based on one of the standardized plants defined during the *Standardize* phase. Project data always has a project-specific context, and is only valid within this single, specific project. On the other hand, master data is independent of any specific project. It is defined once and afterwards it can be reused multiple times in specific projects. Project data is created by instantiation of master data.

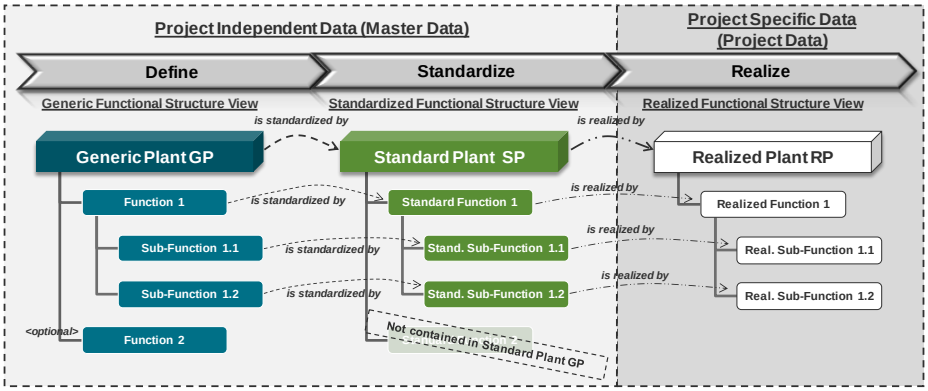

Fig. 1. Engineering of a Plant

Another fundamental characteristic of the object model is the traceability through the complete process. This means, that each of the objects created during the functional engineering process has a link to the object (created in the preceding step) from which it is derived. In our object model, these links are realized by the two relations "*is standardized by"* and "*is realized by"*. *Standard Function 1* on the one hand has been standardized by using the *Generic Function 1* (*is standardized by*). On the other hand, *Standard Function 1* has also been the basis for the instantiation of the *Realized Function 1* (*is realized by*).

3.2 Integrated Engineering based on AML

The *Function-Based Engineering* approach described in the previous section represents a new way of engineering and standardizing industrial production plants from a conceptual point of view. When applying the approach in order to engineer a new production plant from scratch, there are several different technical domains involved in different phases of the process (e.g. mechanical engineering, electrical engineering, software engineering etc.). There are certain dependencies between these domains, which require intensive interdisciplinary collaboration (e.g. the mechanical engineering of the plant influences the electrical engineering, which again impacts the software engineering etc.).

Since the tasks performed in each of the domains are getting more and more complex, the use of software tools supporting these tasks is crucial for state of the art engineering. Within each of the domains there are usually one or more tools in use – with each domain having its own highly specialized set of tools. To enable interdisciplinary collaboration, standardized and automated information exchange between the domains and its software tools is needed. Analyses of current plant engineering companies show that they are still far from having such a standardized and automated information exchange between their domains [7, 8]. Their collaboration is rather characterized by unstructured and manual exchange of information (Fig. 2).

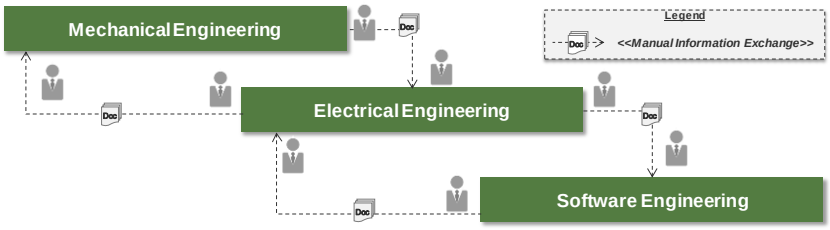

Fig. 2. Sample of a current engineering process

The exchange of information is solely triggered by the personal needs of the involved engineers. If an engineer needs information from another domain, a person-to-person meeting with the engineer of the other discipline is scheduled. During this meeting, the information is exchanged by using unstructured information objects like text documents, spreadsheets etc. [7]. There is no common way of storing and ex-

changing information between all of the disciplines involved. This leads to an inefficient and error prone engineering process due to redundancy and inconsistency of data, slow information exchange and an isolated, territorial mindset [4, 6–8].

In order to solve these problems, our approach was to apply the data integration framework defined by [19]. According to this framework, there are three main characteristics that a system landscape should have in order to efficiently and accurately support engineering processes:

- ***Central Data Platform***: There needs to be a *Central Data Platform* that is the master repository for all relevant data. Alternatively, if such a central system cannot be realized, it would be required to define one system for each information object which acts as the master for it (leading to significantly higher coordination efforts).
- ***Single Point of Communication***: All existing applications use the *Central Data Platform* as their *Single Point of Communication* for all information exchange.
- ***Neutral Data Exchange Format***: Every information exchange between the *Central Data Platform* and its connected applications is performed using a standardized and *Neutral Data Exchange Format.*

In the context of this article, we assume that such a *Central Data Platform* already exists, and that it is able to store and process the *Function-Based Engineering* data model as described above. This *Central Data Platform* is also the *Single Point of Communication* for all applications of the engineering domains, such as mechanical engineering, electrical engineering, software engineering etc. In order to realize the third main characteristic, the *Neutral Data Exchange Format*, we decided to define a concept that enables the XML-based data format AML to serve as our standardized information exchange format (Fig. 3). This concept will be described in more detail in the next section.

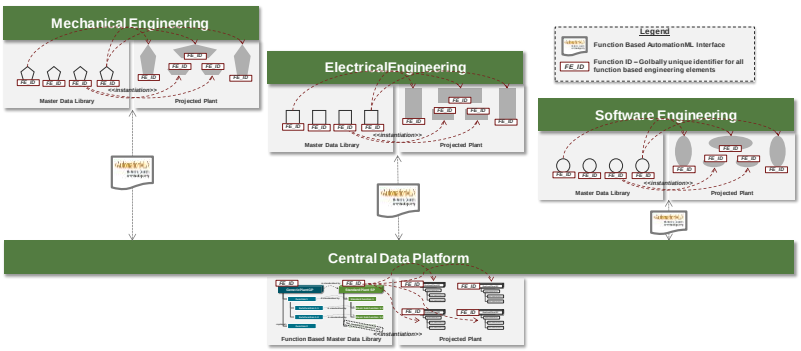

Fig. 3. Integrated engineering concept

3.3 AML Concept

The major requirement that the exchange format must fulfill is the ability to store the *Function-Based Engineering* data model and transfer it between applications without

any data being lost. Additionally, it would be beneficial if the format allowed data transfers to be split up into two independent parts, the master data transfer and the project data transfer, without losing the references between the project data and its associated master data.

With the AML exchange format there is a standardized and neutral format available, which is able to fulfill these requirements. AML was chosen because it is a combination of well-established technologies and capable of describing topology, geometry, kinematics as well as logical information about certain objects. Since the *Function-Based Engineering* framework is focused on the standardization of plant structures in the first place, our concept is only focusing on the topology information part. Geometry, kinematics and logic information are presently excluded from this concept, but may be included at a later point in time, e.g. when adding the manufacturing process of the plants.

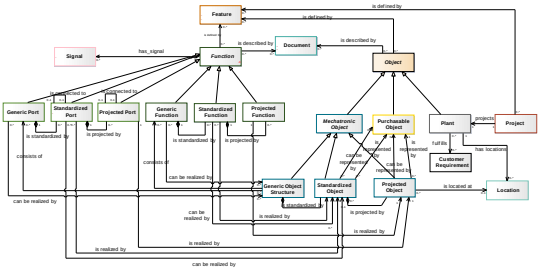

Fig. 4. Data model to be represented using AutomationML

The data model which needs to be exchanged using AML consists of a total of 20 different classes, various relationships (1:1, 1:n and n:m) as well as inheritances between some of the classes. This data model is based on the object model defined by [18] and has been enriched by additional objects to be able to represent all information relevant for the function-based engineering approach (Fig. 4).

To be able to represent this data model using AML, some basic rules needed to be defined in order to ensure that all of the relevant information can be exchanged without any loss. If not explicitly stated otherwise, these rules are extensions to the AML features as they are described in the AML documentation. This set of rules can be split into various categories:

Basic Definitions. The basic definitions determine the structure of the AML interfaces, and which objects are defined in which part of the AML (e.g. *SystemUnitClassLibrary*, *RoleClassLibrary* etc.). The different object types (e.g. *GenericFunction*, *StandardFunction* etc.) and roles (e.g. conveyor, control cabinet) are specified, and naming conventions are defined. Additionally, we developed a concept to enable the representation of 1:n and n:m relationships between different AML classes.

Separation of Master Data and Project Data. As already stated above, it would be favorable if master data and project data could be transferred separately from one another (Fig. 5). This would have multiple positive effects. The most important being

a reduction in interface traffic and at the same time a significant improvement in the interface performance.

When combining master data and project data into one single interface, each transfer of a projected object would also require all related master data information to be transferred at the same time, to ensure that the target system also knows all of the relevant information. If multiple projected objects are based on the same master data, the interface would generate lots of traffic overhead by redundantly transferring the same master data objects again and again.

The problems mentioned above could be resolved by separately transferring master data and project data. Each time a master data object is updated and released for further use in specific projects, this updated master data object is immediately synchronized with all other relevant systems. This process ensures that each system knows all of the master data objects at any time. When transferring a specific project in such a scenario, the transfer can be limited to the relevant projected objects. Prerequisite for such a scenario is that each master data object has a unique identifier, which is valid and available throughout all involved systems.

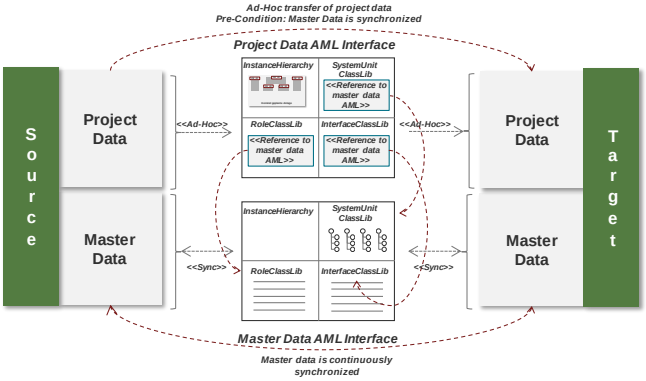

Fig. 5. Separation of master data and project data

The usage of AML allows master data and project data to be separately transferred by supporting references from one AML document to another one. In our approach, we defined, that the master data transfer contains all information stored in the *SystemUnitClassLib*, the *RoleClassLib* and the *InterfaceClassLib*. This ensures that all master data relevant information can be synchronized between the systems. When transferring project data between different systems, the AML document only contains the *InstanceHierarchy*, where the projected elements are represented by *InternalElements*. The reference to the corresponding master data object is created using the *ExternalReference*-option of AML. By using this *ExternalReference*, any information stored in another AML document (in this case the master data document) can be referenced.

Representation of Cardinalities. AML has no out of the box support for the representation of cardinalities. In order to be able to define optional functions or to support variant management (e.g. to be able to define a logic like: *Plant = Function 1 AND*

(Function 2 OR Function 3)) a concept for a representation logic for cardinalities has been specified. This concept allows elements to be defined as being optional, specific cardinalities (e.g. occurrence of exactly two), alternatives between elements (e.g. *either Function 1 OR Function 2*), nested expressions (e.g. either Function 1 OR (Function 2 OR Function 3)) and representations of AND, XOR, OR and n-out-of-m operators.

Representation of Constraints. In certain cases it may be necessary to restrict the values that a specific attribute of an object is allowed to have. AML supports this out of the box by offering a constraint concept. This concept allows attribute values to be restricted to specific values (e.g. "5"), value ranges (e.g. between 5 and 10) or regular expressions for any given attribute. In our approach we applied the constraint concept with only small adjustments.

Mapping of Attributes with the same Semantic. It must be possible to compare the semantic meaning of two attributes. *Function 1* could have an attribute called *Height*. *Function 2* has an attribute called *Absolute Height*. Although the names of the attributes are different, they could describe the same feature. To verify this, each attribute is assigned a semantic reference to an attribute of the *eCl@ss* catalog. *eCl@ss* is a branch-independent catalog for the standardized classification of products and services [20]. If two attributes refer to the same *eCl@ss* attribute, it can be assumed that the semantic meaning of them is identical.

Representation of Ports and Interfaces. Ports and interfaces are used to connect one or more functions with each other. AML has out of the box support for this, using *PortConnector* objects, which can be connected with one another using *InternalLinks*. This functionality is needed to define interconnections between different elements of an engineered plant.

Representation of Documents. To be able to fully document the engineered plants, it is necessary to have the ability to add certain documents to any kind of object available. To enable this, we extended the AML out of the box *ExternalDataConnector* by a *DocumentationInterface*. Using this interface, it is possible to attach any number of documents to an object. Additionally, it is possible to add semantic information to the documents by classifying each of the documents, e.g. by using the standardized document classes and document types of the IEC 61355 norm [21].

Support of the Plant Realization Process. To be able to support the exchange of information during the complete function-based engineering process with its process steps *Define*, *Standardize* and *Realize*, a concept needed to be defined that allows this process to be represented using AML. To achieve this we defined a procedure that allows the representation of each process. For each object represented using AML, full traceability is needed with regards to the process. This means that every projected object has a reference to the standard object that it is based on and each of the standard objects has a reference to its generic object.

Using this set of basic rules as defined above, it is possible to transfer all information relevant for the *Function-Based Engineering* using AML. This enables AML to become the standardized information exchange format for all application interfaces used during the engineering process.

4 Case Studies

In order to prove the feasibility of the AML-based integration approach for the function-based standardization framework, two case studies have been performed. Both have been set up in cooperation with an international IT consulting company, which is also involved in plant engineering projects. The first case study was a greenfield project and has been performed in an environment that is independent of any particular sector. The second case study has been set up in an existing process and system landscape (i.e. brownfield) of an automotive company to be able to evaluate the concept in a sector-specific setting. Table 2 shows more details about organizational environment, company size, duration and stakeholders involved in the case studies.

Table 2. Case study details

Case study	Organizational environment	Company size	Duration	Stakeholders involved
Sector-independent	Innovation project at an IT consulting company	1,200 employees	14 months	Technology and domain experts, architects and developers
Automotive-specific	Customer project at an automotive enterprise	270,000 employees	8 months	Head of engineering, technology and domain experts

4.1 Sector Independent Case Study

During the sector-independent case study, three sub-steps of the plant engineering process for conveyor systems have been covered: *Plant Configuration*, *Detail Engineering* and *Offer Generation* (Fig. 6).

The required *Central Data Platform* was realized using a PLM system (Siemens Teamcenter). The *Function-Based Engineering* data model has been implemented in this PLM system. During the whole plant engineering process, this system is the single point of contact for all other systems used in the different process steps. AML was chosen to be the standardized data exchange format with the central platform. Since Teamcenter does not have any AML capabilities itself, customized AML import and export functions had to be implemented in the PLM system. This implementation was realized using the concept and basic rules described above (including the separated master data and project data transfer).

During the *Plant Configuration* step, a *Configuration System* (own development) was used to support the process. In this system, the conveyor system is configured. First, the standardized master data functions (e.g. straight conveying function, 90 degree conveying function) are placed in a virtual location. Second, all of these functions are individually configured based on the customers' requirements (e.g. motor position, speed etc.). Finally, once the configuration has been completed, the configured conveyor system is transferred via AML to the PLM system and saved as a *Realized Plant* in the project data. This is done by implementing an automated AML export function for project data in the *Configuration System*. Due to technical re-

strictions of the *Configuration System*, an automated master data synchronization was not implemented. During the case study, synchronization was performed manually.

For the subsequent *Detail Engineering* process step, an *Engineering System* (Siemens COMOS) was used to perform detailed engineering steps (e.g. electrical engineering). For this *Engineering System*, AML-based import and export interfaces have been implemented. These interfaces cover the master data transfer as well as the transfer of project-specific objects. The master data transfer has been separated from the project data transfer. Each time, a *Standardized Plant* is released for further use in specific projects, it will be synchronized between the *Central Data Platform* and the *Engineering System* using the master data interface. The project data interface will be initiated, once a *Projected Plant*, which has been configured using the *Configuration System*, was released for *Detail Engineering* in the *Central Data Platform*. The objects are then transferred to the *Engineering System* using AML technology. In the *Engineering System*, details such as circuit diagrams, software modules etc. for the plant are developed. Once these steps are completed, the fully engineered plant is transferred back to the *Central Data Platform* using the project data interface. In the next process step, a standardized offer document is generated for the customer based on the results of the preceding steps. This offer is generated in the *Offer Generator* system. In our case study scenario, this system was integrated into the *Central Data Platform,* which means that in this case no interface had to be implemented. Once the fully engineered *Projected Plant* is released for offer approval, the offer document is automatically generated.

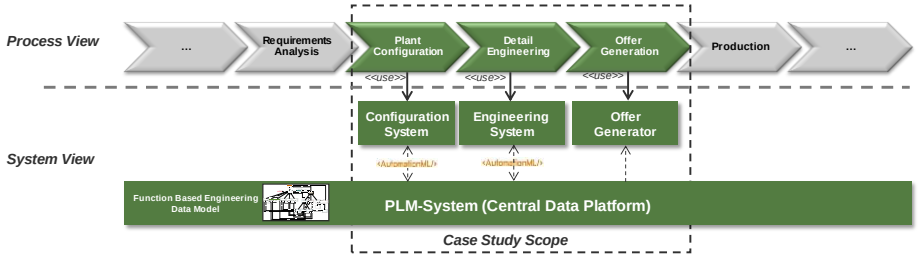

Fig. 6. Process and system overview - sector-independent case study

With this case study we were able to show that the *Function-Based Engineering* data model can be represented using the AML concept described above and that it can be used as a standardized data exchange format for all master data and project data relevant information throughout multiple systems. When using this standardized data exchange in combination with a *Central Data Platform,* it is possible to realize a highly integrated engineering process with a very high degree of automation between the process steps and a high data quality throughout the whole engineering cycle. An integration of additional systems for other process steps (e.g. *Production*) can be realized very easily and without having an impact on the existing landscape.

4.2 Automotive-Specific Case Study

During the automotive-specific case study, two sub-steps of the plant engineering process for automotive production plants were covered: *Production Planning* and *Virtual Commissioning* (Fig. 7). Before the case study was performed, the scenario was characterized by many manual data exchange tasks. The results of the *Production Planning* step have been extracted manually from the *Production Planning System*. Then a manual information exchange was initiated between the planner and the colleague responsible for the *Virtual Commissioning*. The information was exchanged in unstructured meetings or based on agreements and reports. During the *Virtual Commissioning* step, the information was manually entered into a *Target Import File* (Microsoft Excel file), which could be used to import the data into the *Virtual Commissioning System* and continue with the process. This scenario led to high manual efforts and problems regarding data redundancy and data consistency.

The goal of the case study was to automate the data exchange between the *Production Planning System* and the *Virtual Commissioning System* as far as possible. Since the systems were in productive use during the case study, the following constraints did apply: It was not permitted to modify databases and functionalities of the systems involved as well as of the *Target Import File* for the *Virtual Commissioning System*.

After an initial analysis of the situation, an AML-based master data library was built up and defined as the leading system for all master data information relevant for the *Function-Based Engineering*. This library was realized according to the set of basic rules as defined above. At a later point in time, this library may be extended step-by-step towards the *Central Data Platform* for all engineering-relevant data. Then an export function for the *Production Planning System* was realized to export a *Projected Plant* using AML. To be able to combine the project data information from the *Production Planning System* with the master data information in the *Function-Based Engineering Library*, a *Mapping File* had to be set up. In this mapping file, each projected data object included in the *Production Planning System* output gets assigned to exactly one master data object in the *Function-Based Engineering Library*. To be able to perform a mapping, all project and master data objects are required to be identified by their own unique identifiers (UID). Once the project data from the *Production Planning* step is mapped to the *Function-Based Engineering Library*, the *Target Import File* is populated based on AML technology. To achieve this, a *Target System Template* was defined to be able to automatically generate the import file and import the information into the *Virtual Commissioning System*.

The interfaces have been defined in such a way that additional systems of other process steps (e.g. simulation, visualization etc.) can be easily integrated into the landscape by simply extending the *Mapping File* and the functionality of mapping the data of the process-specific systems to the *Function-Based Engineering Library*.

The case study has shown that the *Function-Based Engineering* approach based on AML can even be realized in an existing system landscape, where the involved systems are settled and their functionality as well as their data model is fixed. This can be done by initially defining a central master data repository followed by an incremental integration of the involved systems into the repository.

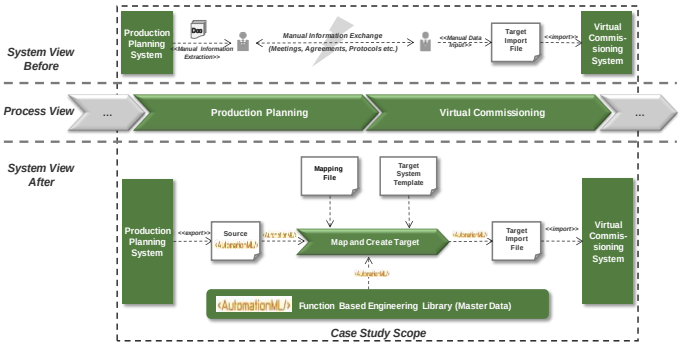

Fig. 7. Process and system overview - automotive-specific case study

4.3 Evaluation

The evaluation of the case study results has shown that such an approach has positive effects on *process cycle times*, *data quality* and *data redundancy (*Table 3). Furthermore, additional systems can be easily integrated without adapting the central repository. A downside of the concept is the relatively *high effort* needed to initially perform the *Function-Based Engineering* in order to be able to populate the central master data repository [18]. The approach described in this article explicitly focuses on the integration of the applications. However, as described in [22], to reach the full potential of the approach, adjustments and optimizations of the existing process landscapes are inevitable and should be considered in further studies.

Table 3. Case study evaluation

Evaluation Criteria	Tendency	Description
Process Cycle Time	↓	We were able to reach a high degree of automation for the interfaces between the systems in use. This led to a significant reduction in the manual work when transferring information between the process steps. ***Measurement***: Manual work before compared to manual work after case study.
Data Redundancy	↘	Due to the introduction of the central master data repository we were able to reduce the degree of data redundancy. All information that is master data-relevant and does not change during the projects is stored one time only in the repository. ***Measurement***: Number of redundant information before compared to after the case study
Data Quality	↗	Due to the reduced data redundancy rate, the data quality with regards to the master data was higher because this data was always referred back to the repository. Before, the master data needed to be synchronized between the systems by manually entering and adjusting the data. After some time, situations will occur where the same information is available with different values in the different systems and it is unknown which system holds the correct value. After the case studies, the central repository always has the correct value of the information and is accessible at any time. ***Measurement***: Number of inconsistent information before compared to after the case sturdy.
Extensibility	↗	By using a central master data repository in combination with the AML interface, new systems can be integrated very easily (as shown by [23]).

Initial Costs	↑	The initial costs for setting up the master data repository were rather high because the plants needed to be engineered using the Function-Based Engineering framework. This is a one-time effort and is initially needed during the setup phase only. ***Measurement***: Project costs needed to realize the concept.

5 Conclusion

The aim of this article was to develop a concept that can be used to realize standardized application interfaces in plant engineering based on AML in combination with the function-based standardization framework for the plant engineering domain. By briefly describing the *Function-Based Engineering Framework* in the beginning, we demonstrated that by using this framework, a better standardization of industrial plants can be realized. Further, we developed an AML-based approach that enables integrated and function-based engineering throughout the whole engineering process. To achieve this we defined a concept and specified a set of basic rules that need to be realized when implementing our concept. Finally, the case study based evaluation proved that this concept can be applied in any plant engineering context, independent of its specific sector. Further, it can be applied in greenfield as well as in brownfield environments. We were able to prove that it is possible to realize highly integrated engineering processes, thus improving the *degree of automation*, *data quality* as well as *data redundancy*.

The concept towards a better standardization and seamless process integration described in this article has the potential to significantly improve the long term efficiency of plant engineering companies. In order to further establish the concept, some additional research should be performed on this topic. On the one hand, additional case studies should be applied in order to further prove the applicability of the concept. On the other hand, some detailed migration strategies are needed when starting from existing plant and system infrastructures. In order to further automate the information exchange, a concept for online data exchange using AML (e.g. using webservices) should be evaluated. Additionally, the concept could be evaluated towards its applicability in context of the "Industrie 4.0" discussions. Cyber-physical systems could directly access the information stored in the central data management platform in order to support their autonomous decision processes. These are still existing challenges which need to be addressed within future research.

References

1. VDI: VDI 4499 Sheet 1 - Digital Factory - Fundamentals, (2008).
2. Himmler, F.: The Digital Factory - A Reference Process based Software Market Analysis. International Journal of Distributed Systems and Technologies. 5, 17–30 (2014).
3. Maurmaier, M., Stuttgart, U., Dencovski, K., Ag, S., Schmitz, E.: Engineering Challenges – Evaluation concept for engineering tools. atp edition. 1, 50–58 (2008).
4. Bohm, B., Gewald, N., Kohlein, A., Elger, J.: Mechatronic models as a driver for digital plant engineering. ETFA2011. pp. 1–8. IEEE (2011).

5. Jazdi, N., Maga, C., Göhner, P., Ehben, T., Tetzner, T., Löwen, U.: Improved Systematisation in Plant Engineering and Industrial Solutions Business – Increased Efficiency through Domain Engineering. at - Automatisierungstechnik. 58, 524–532 (2010).
6. Löwen, U., Wagner, T.: Modelling of complex technical systems – challenges and experiences in industrial plant business. Tagungsband Mechatronik 2009 (2009).
7. Barth, M., Drath, R., Fay, A., Zimmer, F., Eckert, K.: Evaluation of the openness of automation tools for interoperability in engineering tool chains. Proceedings of the 2012 IEEE 17th Conference on Emerging Technologies & Factory Automation (ETFA) (2012).
8. Drath, R., Barth, M.: Concept for interoperability between independent engineering tools of heterogeneous disciplines. Etfa2011. 1–8 (2011).
9. Mobley, K.: Plant Engineer's Handbook. Woburn: Butterworth-Heienmann (2001).
10. Fuchs, J., Feldmann, S., Vogel-Heuser, B.: Modularität im Maschinen- und Anlagenbau - Analyse der Anforderungen und Herausforderungen im industriellen Einsatz. Entwurf Komplexer Automatisierungssysteme EKA. pp. 307–316 (2012).
11. Feldmann, S., Fuchs, J., Vogel-Heuser, B.: Modularity, variant and version management in plant automation – future challenges and state of the art. 12th International Design Conference DESIGN. pp. 1689–1698 (2012).
12. Ji, Y., Borrmann, A., Beetz, J., Obergrießer, M.: Exchange of Parametric Bridge Models Using a Neutral Data Format. Journal of Computing in Civil Engineering. 27, 593–606 (2013).
13. IEC 62714-1: Engineering data exchange format for use in industrial automation systems engineering (AutomationML) Part 1 - architecture and general requirements, (2012).
14. Schleipen, M., Schenk, M.: Intelligent environment for mechatronic, cross-discipline plant engineering. ETFA2011. pp. 1–8. IEEE (2011).
15. Luder, A., Schmidt, N., Helgermann, S.: Lossless exchange of graph based structure information of production systems by AutomationML. 2013 IEEE 18th Conference on Emerging Technologies & Factory Automation (ETFA). pp. 1–4. IEEE (2013).
16. Lüder, A., Hundt, L., Keibel, A.: Description of manufacturing processes using AutomationML. 2010 IEEE 15th Conference on Emerging Technologies & Factory Automation (ETFA 2010). pp. 1–8. IEEE (2010).
17. Makris, S., Alexopoulos, K.: AutomationML server - A prototype data management system for multi disciplinary production engineering. Procedia CIRP. 2, 22–27 (2012).
18. Himmler, F., Loy, H., Ostapovski, V., Amberg, M.: Function Based Engineering - A Standardization Framework for the Plant Engineering Domain. In: Kundisch, D., Suhl, L., and Beckmann, L. (eds.) Tagungsband Multikonferenz Wirtschaftsinformatik 2014 (MKWI 2014). pp. 404–416. , Paderborn (2014).
19. Himmler, F., Amberg, M.: Data Integration Framework for Heterogeneous System Landscapes within the Digital Factory Domain. Procedia Engineering. 69, 1138–1143 (2014).
20. eCl@ss e.V.: eCl@ss - Overview, http://www.eclass.de/eclasscontent/standard/overview.html.en.
21. IEC 61355-1: Collection of standardized and established document kinds, (2008).
22. Himmler, F.: Function Based Requirements Engineering and Design – Towards Efficient and Transparent Plant Engineering. In: Al., G.F.I. et (ed.) SOFSEM 2015: Theory and Practice of Computer Science. pp. 436–448 (2015).
23. Himmler, F.: A Process Driven Data Integration Framework - Towards Integrated System Landscapes for the Digital Factory. Proceedings of the European, Mediterranean & Middle Eastern Conference on Information Systems (EMCIS). pp. 1–16. , Doha, Qatar (2014).