A SysML Profile for the Standardized Description of Processes during System Development

Lasse Beers*, Hamied Nabizada*, Maximilian Weigand*, Felix Gehlhoff*, Alexander Fay*†

*Institute of Automation Technology
Helmut Schmidt University, Hamburg, Germany
{lasse.beers, hamied.nabizada, maximilian.weigand, felix.gehlhoff, alexander.fay}@hsu-hh.de

†Chair of Automation Technology
Ruhr University, Bochum, Germany
alexander.fay@rub.de

***Abstract*—A key aspect in creating models of production systems with the use of model-based systems engineering (MBSE) lies in the description of system functions. These functions should be described in a clear and standardized manner.**

The VDI/VDE 3682 standard for Formalised Process Description (FPD) provides a simple and easily understandable representation of processes. These processes can be conceptualized as functions within the system model, making the FPD particularly well-suited for the standardized representation of the required functions. Hence, this contribution focuses on the development of a Domain-Specific Modeling Language (DSML) that facilitates the integration of VDI/VDE 3682 into the Systems Modeling Language (SysML). The presented approach not only extends classical SysML with domain-specific requirements but also facilitates model verification through constraints modeled in Object Constraint Language (OCL). Additionally, it enables automatic serialization of process descriptions into the Extensible Markup Language (XML) using the Velocity Template Language (VTL). This serialization enables the use of process modeling in applications outside of MBSE. The approach was validated using an collar screwing use case in the major component assembly in aircraft production.

***Index Terms*—SysML Profile, Model-Based Systems Engineering, Domain-Specific Modeling Language, VDI/VDE 3682, Formalised Process Description**

I. Introduction

Approaches of model-based systems engineering (MBSE) enable system architects to respond more quickly and effectively to the numerous changes in the requirements that occur during the development process [1]. Consequently, modeling and management of production resources and planning of resource allocation are becoming increasingly important for manufacturing companies [2].

In the realm of MBSE, it is a well-established practice to investigate the required functions for the system model and capture them in a Blackbox and subsequently providing detailed specifications within a Whitebox [3]. The Whitebox includes the definition of the inputs and outputs of functions. The way these inputs and outputs are exchanged in between functions as well as the sequential order of functions are called functional relationships and are also part of the Whitebox definition. The key experts for defining functional relationships are typically process engineers. Although these engineers possess the necessary expertise, they are often familiar only with their own descriptive tools of process modeling and not with the widely used modeling language Systems Modeling Language (SysML) and its associated software tools in the context of MBSE.

One such descriptive tool for process modeling is the Formalised Process Description (FPD) according to VDI/VDE 3682 [4], which is also easily understandable for other engineers and even non-technical personnel. The FPD precisely defines relationships between in- and output of interrelated processes. These processes can be understood as functions in the system model and therefore the FPD is highly suitable for the modeling of the Whitebox. At the same time, by considering the incoming and outgoing products, as well as the resources required for process execution, it establishes a solid foundation for addressing the prevalent connection between product, process, and resource (PPR) in the manufacturing industry during the modeling of production systems [5].

The aim of this paper is, therefore, to demonstrate how modeling according to the VDI/VDE 3682 can be achieved within the tools in the field of MBSE. For this purpose, a SysML profile has been developed and is presented in this contribution as a Domain-Specific Modeling Language (DSML) to model system functions. This profile can be utilized in modeling system functions to depict the desired behavior and corresponding functional inputs and outputs. Simultaneously, it reduces the complexity of modeling the functional Whitebox while increasing the standardization of the system model. Modeling according to standards enhances understanding and acceptance among participants while facilitating the establishment of universally applicable formal rules, simplifying model verification.

The following sections are structured as follows: In Section II, the fundamentals of this approach are explained.

This research in the iMOD project is funded by dtec.bw – Digitalization and Technology Research Center of the Bundeswehr. dtec.bw is funded by the European Union – NextGenerationEU.

Section III provides an overview of existing approaches to applications using the FPD and modeling system functions in MBSE. In Section IV, the DSML for the VDI/VDE 3682 is introduced, and in Section V, it is applied using an example from aircraft production. The paper concludes with a summary and outlook in Section VI.

II. Fundamentals

A. VDI/VDE 3682 Formalised Process Description

The VDI/VDE 3682 standard [4] describes the conception and use of the FPD. It is a universal tool for the description of all types of processes, both technical and non-technical. All process-related information can be clearly and systematically communicated, reused, and progressively detailed throughout the entire lifecycle of an automated system.

For the graphical modeling of a process, the standard defines a set of permissible symbols and their relationships. State-describing symbols (Product, Energy, Information) are connected to a Process Operator by directed edges (flow connections). The Process Operator describes a technology-neutral transformation of a state ante to a state post. The technical realization of this Process Operator is represented by a usage relationship between the Process Operator and a Technical Resource. To delineate the process from its external environment, a System Boundary is defined around the modeled process. The elements Product, Energy, or Information located on this System Boundary can be understood as inputs or outputs and thus indicate the interactions of the system with its environment. Within the System Boundary, individual process steps can be further specified through the decomposition of the respective Process Operators. A small abstract example featuring the graphical elements is outlined in Figure 1.

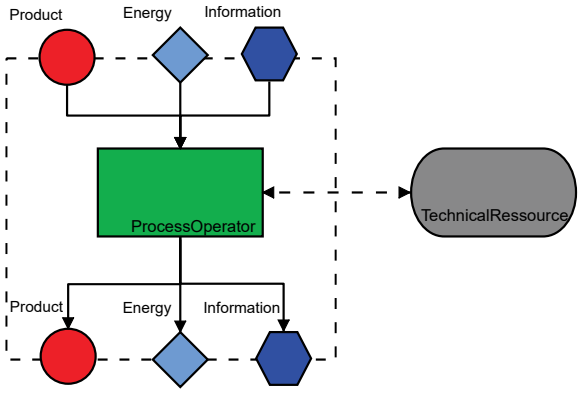

Fig. 1: Graphical Elements of the FPD

In addition, both parallel and alternative process flows can be represented without expanding the symbol set. In graphical modeling, a parallel flow is represented by a partially shared flow, while alternative flows are represented by straight flows.

The graphical notation of the FPD is supported by an underlying object-oriented information model [6]. The class diagram of this information model is shown in Unified Modeling Language (UML) notation in Figure 2 and forms the basis for the developed profile of this contribution, which is introduced in Section IV.

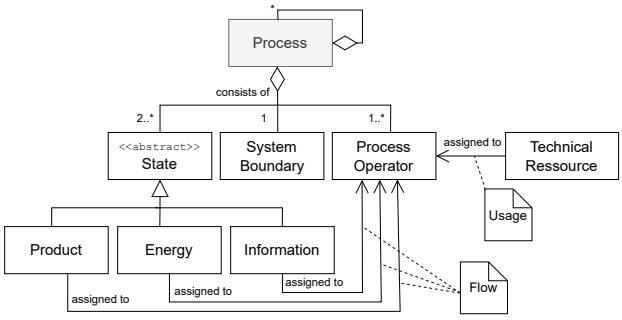

Fig. 2: Class Diagram of the FPD. Adapted from [6].

Furthermore, the standard VDI/VDE 3682 outlines how the identification of individual elements and the description of their features can be modeled. Identification includes, for example, a unique ID, a short name, and a long name [6].

B. Domain-Specific Modeling Language (DSML)

DSMLs enable developers to reconstruct and apply domain-specific concepts precisely in a suitable modeling language. The use of such languages contributes to enhancing the quality of modeling, as inconsistent modeling can be avoided to some extent through the clear syntax and semantics of DSMLs [7].

UML offers a wide range of possibilities to model knowledge in the form of various elements and diagrams. However, the variety offered is often not sufficient for modeling domain-specific knowledge. In order to meet the domain-specific modeling challenges, a lightweight or a heavyweight DSML can be developed. Modifying the existing UML metamodel or even developing an entirely new metamodel is referred to as a heavyweight approach [8]. If the existing UML metamodel is taken as the base without any modifications and is extended with domain-specific requirements or notations, it is referred to as a lightweight approach [9].

Specially for lightweight approaches, the UML profiling mechanism was developed, allowing an extension of the metamodel. The most well-known example of such a UML profile is SysML. It extends UML with additional elements and diagrams and has become one of the most established languages for modeling complex systems [9]. A key advantage of using this mechanism is the standardization and the associated reusability of a profile. Through the standardized exchange format called XML Metadata Interchange (XMI) established by the Object Management Group, a developed profile can be easily imported or exported, making it accessible to other users. The central component of the profile is the stereotype. It is a distinct metaclass within UML that can extend or restrict other existing metaclasses by adding additional meta-attributes. A stereotype can also inherit properties from multiple metaclasses or other stereotypes. Furthermore, the visual appearance of the altered stereotype can be modified. The extension of a stereotype through meta-attributes

is implemented by creating Tagged Values (Tags). Tags are attributes specifically created for a stereotype.

The simplest form of Tags are regular attributes. When added, every element of the modified stereotype acquires this property. However, more complex Tags can also be developed. A commonly used example is the Derived Property, which is a characteristic whose value is defined by its relationship with other elements. The value is dynamically calculated, ensuring it automatically updates during model adjustments [10]. Derived Properties can be defined by a simple relationship, a linked meta chain, or by other types of code. These are implemented within the element `Customization`. However, Customization also provides a way to limit the use of a stereotype. This includes the definition of Connection Rules or Model Initialization. Connection Rules, such as `TypeOfSource`/`TypeOfTarget`, restrict the scope of elements that can be the source of a relationship to elements belonging to a certain stereotype. An example of this is that a usage relationship (the dotted line between the process operator and the technical resource in Figure 1) can only be used to connect the corresponding stereotypes. Similarly, visibility and usage within other elements can be manipulated. However, these limitations defined through the customization are not sufficient to impose all possible restrictions.

Ths fact leads us to another essential tool for restricting a profile which is called Constraints. These can be developed independently of the profile and later added to one or more stereotypes. Constraints can be modeled in Object Constraint Language (OCL) or in programming languages like Java. Within the Constraint, complex relationships can be defined [11]. For example, it can be defined that each FBD diagram must have at least one input state and one output state. These states are recognized by being on the system boundary.

The development of new diagrams, specifically tailored to the domain of DSML, is also possible. Completely new diagrams can be developed, or existing UML/SysML diagrams can be used as a foundation, which is then extended or restricted according to the specific needs. For this purpose, the newly developed stereotypes are defined as usable elements for the corresponding diagram.

Furthermore, various software tools provide additional functionalities to support the individual creation of a DSML [12]. For example, the software tool "Magic Systems of Systems Architect (MSoSA)", developed by Dassault Systèmes, provides advanced customization possibilities, such as implementing custom logic with Java code.

III. State of the art

In this section, approaches from VDI/VDE 3682 that utilize the FPD to create various model types are presented. Concurrently, the current status of potential serialization efforts is outlined. The second part of this section introduces various MBSE approaches that encompass the modeling of system functions and the Blackbox and Whitebox.

A. Usage and Serialization of the VDI/VDE 3682

Since the publication of the VDI/VDE 3682, it has been gaining increasing significance due to its diverse applications. In [13], an Ontology Design Pattern is introduced to map the information model of the standard, offering a reusable pattern for process descriptions using ontologies in Web Ontology Language (OWL). A possible mapping of the FPD into Automation Markup Language (AML) [14] was presented in [15]. The FPD is also chosen as a suitable means of description in capability models (e.g., in [16] or [17]) and digital process twins [18]. The authors of [19] use the FPD as a starting point for the mathematical description of interdependencies among process parameters. In [20], a formal metamodel was developed based on the FPD, enabling consistency checks. The authors of [21] use the FPD as a basis for an intelligent production process planning algorithm. An open-source web-based tool[1] facilitates standard-compliant process modeling [22]. This tool employs a custom serialization format using JSON for storing process descriptions.

The serialization in XML format announced for part 3 of the VDI/VDE 3682 standard has not been standardized yet. However, a proposal for such serialization is provided in [23]. The serialization of the process descriptions described in Section IV-C follows the proposal in [23]. To the best of our knowledge, there is currently no approach that represents the FPD as a DSML based on SysML.

B. Approaches for Modeling System Functions with SysML

SysML stands out as a widely adopted graphical modeling language, establishing itself as the de-facto standard within the realm of MBSE . Renowned for its versatility, SysML provides an extensive range of capabilities, facilitating the specification, analysis, design and verification of intricate systems [24].

The Software Platform Embedded Systems (SPES) Modeling Framework was created within the context of various research endeavors, including SPES2020 [3], SPES XT [25], and CreSt [26]. Its objective is to provide language-independent modeling methods for the design of embedded systems and software systems. These methods have also been implemented in various approaches, such as those presented in [27] and [28], utilizing the SysML modeling language. The framework adheres to an iterative process encompassing the delineation of distinct layers and viewpoints. To capture the functional viewpoint, it proposes a description utilizing Blackbox and Whitebox functions. This approach aims to mitigate complexity by hierarchically structuring functionality from the user's perspective, overseeing functional interactions, and comprehending functional relationships through the capturing and analyzing of interactions between different (sub-)functionalities. Notably, the functions within the functional Whitebox model must collectively exhibit a behavior identical to that specified by the user function in the functional Blackbox model [3].

[1] publicly available at https://demo.fpbjs.net

Based on the knowledge and methods developed within in the context of SPES and CreSt, Hayward et al. introduce a function-centered approach for describing collaborative cyber-physical systems, employing the SysML [27]. The authors emphasize the importance of modeling Blackbox and Whitebox for the function-centric view. Blackbox functions are modeled as a stereotype of the metaclass Activity, representing the functional structure of the system group. Whitebox stereotypes are then utilized to specify the Blackbox, defining the sequential process and assignment to roles. However, the approach does not involve modeling inputs and outputs for the definition of Blackbox and Whitebox. The correctness of the modeling is not guided by corresponding rules or constraints.

Welkiens et al. propose to design the functions of a system using the Functional Architecture for Systems (FAS) method [29]. This method involves grouping the activities of the use case through the identification of functional groups. Subsequently, taking into account both functional and non-functional requirements, the functional architecture is modeled using a SysML Internal Block Diagram. The functional groups are related and connected through corresponding ports. However, this approach quickly leads to a cluttered and complex representation of system functions, making it understandable and implementable only by modeling experts. Additionally, the lack of standardization introduces a considerable degree of interpretation in the modeling process.

Binder et al. outline in their approach [30] the necessity of utilizing the PPR concept in the realm of Systems Engineering, particularly when modeling a production system. The authors have developed a UML metamodel for representing the PPR context and employ process descriptions at the functional level. However, the approach only considers products as inputs for processes and lacks reliance on any standard. Furthermore, no verification mechanisms are introduced.

The modeling approach presented in [28], which is also aligned with the SPES method, incorporates the modeling of both Blackbox and Whitebox aspects within the definition of the functional viewpoint. In this approach, the FPD is used as a basis for modeling. The inputs and outputs of the functions are modeled through State-describing elements prescribed in the standard, namely Information, Product, and Energy. However, simple SysML elements are used for modeling, which prevents guided modeling due to the limitations in the use or verification of the models.

In summary, none of the presented approaches in MBSE allows for the comprehensive modeling of the PPR concept according to a standard. This gap is addressed by the design of a SysML profile for the VDI/VDE 3682 standard[2]. The development of a profile for the VDI/VDE 3682 enables the guided modelling according to the standard and the seamless integration into the entire system development process. This increases the usability for the system engineer. The main advantage of using this profile instead of the classic UML activity diagram is the reusability and the facilitated verification of the model during the modeling process. Reusability is achieved by using a standardized description of system functions. Model verification is achieved by defining OCL constraints and is explained in Section IV-B.

[2]publicly available at https://github.com/hsu-aut/IndustrialStandard-SysML-Profile-VDI3682

IV. DSML FOR VDI/VDE 3682

A. VDI/VDE 3682 Profile

For the development of a DSML for the VDI/VDE 3682, a lightweight approach based on SysML is a suitable option. The rationale behind this lies in the fact that elements from UML and SysML already offer analogous behavioral modeling constructs that readily lend themselves to straightforward extensions. Moreover, this approach facilitates the reuse of established tools, such as constraint verification through OCL, and affords the seamless integration of these lightweight methodologies into extant MBSE workflows, such as [28]. To achieve this, SysML is extended with additional stereotypes to meet the requirements of VDI/VDE 3682. Therefore, this section introduces the elements of the DSML and explains how they address the specific modeling requirements of this standard. Table I provides an overview of all elements in the DSML along with their corresponding metaclasses, in comparison with the elements of the FPD.

TABLE I: Overview of the elements of the DSML

FPD Element	DSML Element	SysML/UML Metaclass
FPD Diagram	FPD Diagram	Activity Diagram
Process	Process	Activity
System Boundary	Diagram Boundary	-
Process Operator	Process Operator	CallBehaviorAction
Technical Resource	Technical Resource	Class
Usage	Usage	Dependency
Product	Product	ActivityParameterNode
	Intermediate Product	ObjectNode
Energy	Energy	ActivityParameterNode
	Intermediate Energy	ObjectNode
Information	Information	ActivityParameterNode
	Intermediate Information	ObjectNode
Flow	Flow	ActivityFlow
		ObjectFlow
Parallel Flow	ForkNode	ForkNode
	JoinNode	JoinNode
Alternative Flow	MergeNode	MergeNode
	DecisionNode	DecisionNode
Identification	Identification	Attribute
Characteristics	Characteristics	Attribute

Following the VDI/VDE 3682 standard, every Process is defined by a single System Boundary (cf. Figure 2), delineating the process from its external environment. This can be equated with the diagram boundary in the DSML. The inputs and outputs of the system align with the elements situated on this diagram boundary, consistent with the standard.

A process, according to the standard, can be understood as a description of a system's behavior. Therefore it is appropriate to use a behavior diagram of SysML as a fundamental diagram type. Particularly suitable for this purpose is an `Activity Diagram`, as within this diagram, both inputs and outputs

can be described, and furthermore, decomposition of Process Operators are possible. Based on the `Activity Diagram`, a standalone diagram type was created, on which only the elements of the FPD which are defined in the profile can be used. This leads to more guided modeling, helping to avoid modeling errors.

Since each FPD Diagram requires at least one Process element, a stereotype has been created for this element which extends the metaclass `Activity`. This element is always directly assigned to the FPD Diagram. Additionally, a `Derived Property` has been created to ensure that the start and end State elements are displayed as additional `Attributes`. These attributes are subsequently used to verify whether each process has at least one start and one end State.

As shown in Figure 3 the Process Operator is represented by a stereotype of the metaclass `CallBehaviorAction`, which can be decomposed into another FPD diagram. Within the element `Customization`, the `Possible Owners` are defined, restricting the usage of this stereotype. The derived properties refer to the State-describing elements entering or exiting the Process Operator, as well as to the Technical Resource linked to the Process Operator through the Usage relationship. This Usage relationship is derived from the metaclass `Dependency`. These are also utilized within the Constraints defined in OCL to verify whether each Process Operator has at least one input and one output connected by a flow. This Flow is inherited from `ActivityFlow` and `ObjectFlow`, ensuring the representation of the sequential order and the proper passing of all information. Through additional `Derived Properties`, it is ensured that all information can be passed on during the decomposition.

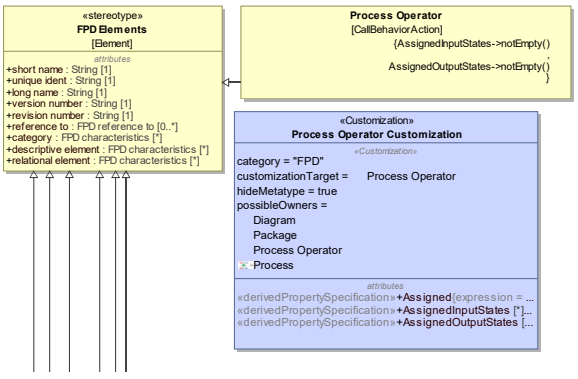

Fig. 3: Stereotype and Customization of the Process Operator

The Technical Resource element is implemented through a corresponding stereotype, which belongs to the metaclass `Class` and also inherits the properties of the SysML stereotype `System`. Since SysML excludes the possibility of modeling elements outside diagram boundaries, the Technical Resource elements must be modeled within the FPD diagram. This violates the standard, but it is not a serious breach.

The State-describing elements (Product, Energy, Information) that lie on the System Boundary have outputs only when they are introduced into the system or inputs only when they are leaving the system. However, the elements within the System Boundary must have both inputs and outputs. To formulate appropriate constraints for adhering to these rules, different stereotypes for the two variants have been modelled. The State-describing elements on the System Boundary, therefore, receive the same name as those from the standard and are implemented as stereotypes of the metaclass `ActivityParameterNode`,that are used at the beginning and end of activities to accept inputs and provide outputs. Those within the System Boundary receive the prefix Intermediate in their nomenclature and are implemented as stereotypes of the metaclass `ObjectNode`. This differentiation enables targeted querying and constraint modeling.

The elements Identification and Characteristics are implemented using additional stereotypes, and all the subordinate FPD elements are represented by `Attributes`. These elements are decomposed by nesting the corresponding attributes and stereotypes. In the profile, a parent stereotype called FPD Elements has been introduced. All other FPD stereotypes inherit the attributes of this stereotype through the generalization relationship.

Within the FPD, it is possible to model parallel and alternative flows. Parallel flows are implemented in the profile using the elements `ForkNode` and `JoinNode`, which are already part of the SysML specification. Alternative flows are implemented through the SysML elements `DecisionNode` and `MergeNode`. This approach indeed expands the modeling elements of the standard, but it allows for a meaningful grouping capability of input and output elements, as proposed in [23].

B. Constraint Modeling

The meta model of the VDI/VDE 3682 imposes certain constraints on the use of its elements. Examples of these are:

- Each process must have at least one Process Operator.
- The state-describing elements Product, Information, and Energy must always be associated with a Process Operator.
- Each process must always have at least two state-describing elements.

These are reflected in the SysML profile and can be verified through formal rules. However, certain aspects of the UML class diagram of the FPD standard are unclear. For example, the cardinality of the relationship between States and processes indicates that a process must consist of at least two State-describing elements. Nevertheless, only the textual description clarifies that this involves at least one element describing the input and at least one element describing the output of the process.

Additionally, it is useful for implementations to be able to distinguish whether the State-describing elements are located on or within the system boundary. Only in this way, formal rules can be defined to check whether the modeling rule regarding the connection of a State-describing element to a Process Operator is fulfilled. Through this approach, specific

queries can be directed at the inputs and outputs of the corresponding elements to verify the existence of this connection. A State-describing element within the system boundary must have connections to a Process Operator as both an input and an output. For the State-describing elements on the system boundary, this connection must only be fulfilled either at the input or the output.

These and additional constraints have been implemented using OCL to enable standard-compliant modeling.

Listing 1 exemplifies a constraint written in OCL. This constraint is associated with the Flow Element and defines that this element cannot simultaneously have an element of the Information, Energy, and Product stereotypes as both source and target. This constraint prevents two State-describing elements from being connected together.

Listing 1: Constraint for Flow Stereotype defined in OCL

```
context Flow inv FlowsSourceAndSource
((appliedStereotype->exists(name='Flow'))
and not (source->exists(out | out.oclIsTypeOf(Information) or out.oclIsTypeOf(
    Product)
or out.oclIsTypeOf(IntermediateInformation)or out.oclIsTypeOf(IntermediateProduct)
or out.oclIsTypeOf(IntermediateEnergy)or out.oclIsTypeOf(Energy))
and target->exists(trg | trg.oclIsTypeOf(Information) or trg.oclIsTypeOf(Product)
or trg.oclIsTypeOf(IntermediateInformation)or trg.oclIsTypeOf(IntermediateProduct)
or trg.oclIsTypeOf(IntermediateEnergy) or trg.oclIsTypeOf(Energy))))
```

C. Automated Code Generation

However, XMI is specifically tailored for exchanging metadata between modeling tools, such as SysML modeling tools and other software engineering. As highlighted in Section III-A, there is a need to utilize process descriptions in other description languages and interchange formats, such as AML or OWL, in a standards-compliant manner. One approach to achieve this is the use of an established and versatile interchange format like Extensible Markup Language (XML).

The upcoming part 3 of the VDI/VDE 3682 envisions a unified XML schema for representing the standard. A promising proposal for a standardized XML representation has already been published with the approach outlined in [23]. This approach is used here for the automatic generation of a formalized representation of the FPD in XML.

To facilitate export into the XML format, templates written in Velocity Template Language (VTL) can be utilized. These templates enable the automatic generation of the interchange format from the existing SysML model using the Apache Velocity Engine.

Listing 2 displays a snippet from the VTL code. The shown excerpt searches the Node attribute of all process elements. If elements are found that exhibit the stereotypes Product, Information, or Energy, a XML element `state` with the corresponding attributes is generated for each of these elements. Additionally, references are made to the ID of the Process Operator linked with the State-describing Element and the IDs of the corresponding Flows. The template's defined structure facilitates precise mapping to the proposed XML representation of the FPD in [23].

Listing 2: Snippet of the VTL Template

```
#foreach ($State in $Process.Node)
#if($State.AppliedStereotype.size()>0)
#if($State.AppliedStereotype[0].Name == "Product" || $State.AppliedStereotype[0].
    Name == "Information" || $State.AppliedStereotype[0].Name == "Energy")
        <state stateType="$State.AppliedStereotype[0].Name">
            <identification uniqueIdent="$State.ElementID" shortName="$State.
                ShortName" longName="$State.LongName" versionNumber="$State.
                versionNumber" revisionNumber="$State.revisionNumber">
                <references></references>
            </identification>
            <characteristics>
            </characteristics>
            <assignments>
#foreach($AssignedID in $State.assignedToProcessOperator)
                <assigned id="$AssignedID.ElementID" />
#end
            </assignments>
            <flows>
#foreach($OutgoingFlow in $State.Outgoing)
                <flow>
                    <exit id="$OutgoingFlow.ElementID" />
                </flow> #end
#foreach($IncomingFlow in $State.Incoming)
                <flow>
                    <entry id="$IncomingFlow.ElementID" />
```

This export capability allows the approach to be utilized by users who want to model processes according to this standard independently of a Systems Engineering activity, such as for production capabilities [16] or process parameter interdependencies [19]. Furthermore, this work takes another step towards standardizing an XML-based exchange format for the VDI/VDE 3682, as it demonstrates the applicability of the proposal for mapping FPD to XML from [23].

V. Application Example

The profile mechanism introduced in Section IV is utilized and demonstrated using an application example from aircraft production. In this example, a system is developed to screw collars within an aircraft fuselage. The model was implemented using the MSoSA software from Dassault Systèmes. A workflow for the development of an aircraft production system has already been introduced in [28]. This workflow is also used in this application example. During the development of the Functional Viewpoint, the Functional Requirements modeled in the Requirements Viewpoint are initially used to derive the Blackbox functions. An excerpt of the Blackbox functions can be found in Figure 4. Building on this, the individual Blackbox functions are intended to be connected, and a sequential series of steps is defined. For this purpose, the newly developed profile is applied.

The `Activity` elements of the Blackbox functions are assigned to the stereotype Process. Subsequently, an FPD Diagram is created for each of these processes. This is exemplified in Figure 5.

The initial State consists of the Product *Collar* and the Information *Rivet Position*. Additionally, an energy supply must be ensured during the process. The Process Operator *Automated Collar Screwing* transforms this initial State into the end State *Screwed Collar* as the Product and *Thermal Energy* as the Energy output. The inputs and outputs of the system functions can be derived from the system context. A detailed insight into this aspect can be found in [28].

Subsequently, the Process Operator is decomposed into additional sub-processes. In Figure 5, an error was delib-

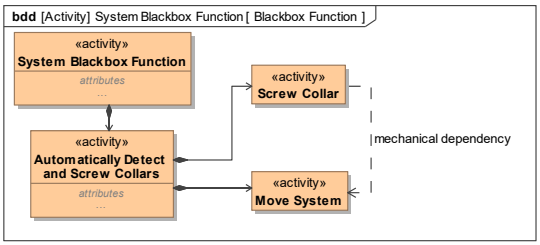

Fig. 4: Blackbox functions of the Collar Screwing System

erately introduced to exemplify error identification. In the example, two state-describing elements were connected (see red arrow). While the Flow stereotype is permitted to have a State stereotype as both source and target types, the OCL constraints prohibit this modeling. The user is alerted to the corresponding error through the error message: “A state must always be assigned a process operator. Linking two states is not permitted”.

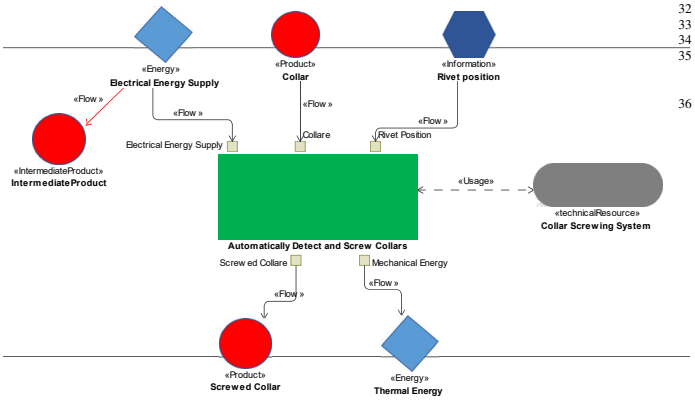

Fig. 5: Implementation Example

After addressing all errors, the process description can be automatically exported to XML following the approach described in Section IV-C and using the internal report wizard of MSoSA. An excerpt of the serialization is exemplified in Listing 3 and demonstrates the mapping of the input elements from Figure 5 to the XML schema proposed by [23]. For clarity, the listing does not display any Characteristics, but they can be created through the profile as mentioned in Section IV-A. Nevertheless, it can be seen that the process model has been transferred to the correct exchange format.

VI. Summary and Outlook

By using the FPD according to VDI/VDE 3682, processes can be structured and modeled using simple means based on the PPR concept. This proves to be particularly effective in Systems Engineering as an instrument to break down the functional Blackbox and transition to a functional Whitebox.

In this context, the utilization of a DSML was motivated and developed as a profile of SysML. In addition to graphical modeling capabilities, this DSML incorporates OCL rules,

Listing 3: Excerpt of the XML serialisation

```
<process id="_2021x_56901f2_1698910860901_408207_17462">
    <systemLimit id="_2021x_56901f2_1698910860895_144044_17461" shortName="Modeling
        Example" />
        <states>
            <state stateType="Energy">
                <identification uniqueIdent="
                    _2021x_56901f2_1698911485391_499621_17572" shortName="
                    Electrical Energy Supply" longName="" versionNumber="
                    revisionNumber="">
                    <references></references>
                </identification>
                <characteristics>
                </characteristics>
                <assignments>
                    <assigned id="_2021x_56901f2_1698911444248_32009_17539" />
                </assignments>
                <flows>
                    <flow>
                        <exit id="_2021x_56901f2_1698911552130_199686_17659" />
                    </flow>
                </flows>
            </state>
            <state stateType="Product">
                <identification uniqueIdent="
                    _2021x_56901f2_1698911586000_254919_17680" shortName="Collar"
                     longName="" versionNumber="" revisionNumber="">
                    <references></references>
                </identification>
                <characteristics>
                </characteristics>
                <assignments>
                    <assigned id="_2021x_56901f2_1698911444248_32009_17539" />
                </assignments>
                <flows>
                    <flow>
                        <exit id="_2021x_56901f2_1698911938863_856843_17815" />
                    </flow>
                </flows>
            </state>
            <state stateType="Information">
                <identification uniqueIdent="
                    _2021x_56901f2_1698911694625_303176_17766" shortName="Rivet
                    position" longName="" versionNumber="" revisionNumber="">
                ...
```

allowing for the verification of modeling errors and inconsistencies. The DSML enables process modeling according to the standard within MBSE tools. The process descriptions can be not only utilized within MBSE tools but also serialized in XML, enabling their application in various contexts. Finally, using a case study from aircraft production [5], it was demonstrated how the DSML facilitates the transition from a Blackbox to a Whitebox.

Future work will be concerned with automating several mechanisms and integrating them into a plug-in using the DSML. This will enhance the guidance and support for process modeling. Additionally, the approach is intended to be applied to further use cases in Systems Engineering to further improve it and show its general applicability. The process modeling is used within the Functional Viewpoint, which was presented in the paper [28] and is based on the SPES method. However, the representation of VDI/VDE 3682 does not only include the process. It also includes the Technical Resource.The Technical Resources described there can be used as elements of the Logical Viewpoint to describe the logical architecture of the system to be developed. These will be further specified in the Technical Viewpoint later on. How to appropriately model the technical resources will be presented in a future paper.

References

[1] J. D'Ambrosio and G. Soremekun, "Systems engineering challenges and MBSE opportunities for automotive system design," in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2017, pp. 2075–2080.

[2] E. M. Sanfilippo, S. Benavent, S. Borgo, N. Guarino, N. Troquard, F. Romero, P. Rosado, L. Solano, F. Belkadi, and A. Bernard, "Modeling Manufacturing Resources: An Ontological Approach," in *Product Lifecycle Management to Support Industry 4.0*, 2018, pp. 304–313.

[3] K. Pohl, H. Hönninger, R. Achatz, and M. Broy, *Model-based engineering of embedded systems: The SPES 2020 methodology*. Springer, 2012.

[4] VDI/VDE 3682-1, "Formalised Process Descriptions - Concept and Graphic Representation," 2015.

[5] F. Gehlhoff, H. Nabizada, M. Weigand, L. Beers, O. Ismail, A. Wenzel, A. Fay, P. Nyhuis, W. Lagutin, and M. Röhrig, "Challenges in automated commercial aircraft production," *IFAC-PapersOnLine*, vol. 55, no. 2, pp. 354–359, 2022.

[6] VDI/VDE 3682-2, "Formalised Process Descriptions - Information model," 2015.

[7] U. Frank, "Domain-Specific Modeling Languages: Requirements Analysis and Design Guidelines," *Domain Engineering: Product Lines, Languages, and Conceptual Models*, pp. 133–157, 2013.

[8] F. Lagarde, H. Espinoza, F. Terrier, C. André, and S. Gérard, "Leveraging Patterns on Domain Models to Improve UML Profile Definition," in *International Conference on Fundamental Approaches to Software Engineering*. Springer, 2008, pp. 116–130.

[9] M. Seidl, M. Scholz, C. Huemer, and G. Kappel, *UML @ Classroom: An Introduction to Object-Oriented Modeling*. Springer, 2015.

[10] MagicDraw, "UML Profiling and DSL 18.1 - User Guide," 2015. [Online]. Available: https://docs.nomagic.com/download/attachments/17667876/MagicDraw%20UMLProfiling%26DSL%20UserGuide.pdf

[11] E. Bousse, D. Mentré, B. Combemale, B. Baudry, and T. Katsuragi, "Aligning sysml with the b method to provide v&v for systems engineering," in *Proceedings of the workshop on model-driven engineering, verification and validation*, 2012, pp. 11–16.

[12] J. Cabot and M. Gogolla, "Object Constraint Language (OCL): A Definitive Guide," in *Formal Methods for Model-Driven Engineering: 12th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM 2012*. Springer, 2012, pp. 58–90.

[13] C. Hildebrandt, S. Törsleff, B. Caesar, and A. Fay, "Ontology building for cyber-physical systems: A domain expert-centric approach," in *2018 IEEE 14th international conference on automation science and engineering (CASE)*. IEEE, 2018, pp. 1079–1086.

[14] R. Drath, *AutomationML: A Practical Guide*. Walter de Gruyter GmbH & Co KG, 2021.

[15] T. Jäger, L. Christiansen, M. Strube, and A. Fay, "Durchgängige Werkzeugunterstützung von der Anforderungserhebung bis zur Anlagenstrukturbeschreibung mittels formalisierter Prozessbeschreibung und AutomationML," *Proceedings of EKA*, pp. 8–10, 2012.

[16] A. Köcher, C. Hildebrandt, L. M. Vieira da Silva, and A. Fay, "A Formal Capability and Skill Model for Use in Plug and Produce Scenarios," in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, 2020, pp. 1663–1670.

[17] A. Köcher, A. Belyaev, J. Hermann, J. Bock, K. Meixner, M. Volkmann, M. Winter, P. Zimmermann, S. Grimm, and C. Diedrich, "A reference model for common understanding of capabilities and skills in manufacturing," *at - Automatisierungstechnik*, vol. 71, no. 2, pp. 94–104, 2023.

[18] B. Caesar, A. Hänel, E. Wenkler, C. Corinth, S. Ihlenfeldt, and A. Fay, "Information Model of a Digital Process Twin for Machining Processes," in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, 2020, pp. 1765–1772.

[19] T. Jeleniewski, H. Nabizada, J. Reif, A. Köcher, and A. Fay, "A Semantic Model to Express Process Parameters and their Interdependencies in Manufacturing," in *2023 IEEE 32nd International Symposium on Industrial Electronics (ISIE)*, 2023, pp. 1–6.

[20] L. Kathrein, K. Meixner, D. Winkler, A. Lüder, and S. Biffl, "A Meta-Model for Representing Consistency as Extension to the Formal Process Description," in *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2019, pp. 1653–1656.

[21] P. Novák and J. Vyskočil, "Digitalized Automation Engineering of Industry 4.0 Production Systems and Their Tight Cooperation with Digital Twins," *Processes*, vol. 10, no. 2, p. 404, 2022.

[22] H. Nabizada, A. Köcher, C. Hildebrandt, and A. Fay, "Offenes, webbasiertes Werkzeug zur Informationsmodellierung mit Formalisierter Prozessbeschreibung," in *Automation 2020 - Shaping Automation for our Future*, 2020.

[23] H. Nabizada, T. Jeleniewski, A. Köcher, and A. Fay, "Vorschlag für eine XML-Repräsentation der Formalisierten Prozessbeschreibung nach VDI/VDE 3682," in *17. Fachtagung EKA - Entwurf komplexer Automatisierungssysteme*, 2022.

[24] OMG, "Systems Modeling Language (SysML™ 1.6)," 2019. [Online]. Available: https://www.omg.org/spec/SysML/1.6/PDF

[25] K. Pohl, M. Broy, H. Daembkes, and H. Hönninger, *Advanced Model-Based Engineering of Embedded Systems: Extensions of the SPES 2020 Methodology*. Springer, 2016.

[26] W. Böhm, M. Broy, C. Klein, K. Pohl, B. Rumpe, and S. Schröck, *Model-Based Engineering of Collaborative Embedded Systems: Extensions of the SPES Methodology*. Springer Nature, 2021.

[27] A. Hayward, M. Rappl, and A. Fay, "A SysML-based Function-Centered Approach for the Modeling of System Groups for Collaborative Cyber-Physical Systems," in *2022 IEEE International Systems Conference (SysCon)*, 2022, pp. 1–8.

[28] L. Beers, M. Weigand, H. Nabizada, and A. Fay, "MBSE Modeling Workflow for the Development of Automated Aircraft Production Systems," in *2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2023, pp. 1–8.

[29] T. Weilkiens, J. G. Lamm, S. Roth, and M. Walker, *Model-Based System Architecture*. John Wiley & Sons, 2022.

[30] C. Binder, P. Hünecke, C. Neureiter, and A. Lüder, "Towards flexible production systems engineering according to RAMI 4.0 by utilizing PPR notation," in *2023 IEEE 21st International Conference on Industrial Informatics (INDIN)*, 2023, pp. 1–6.