



Article

EIPPM—The Executable Integrative Product-Production Model [†]

Dominik Schopper ^{1,*}, Karl Kübler ^{2,‡}, Stephan Rudolph ¹ and Oliver Riedel ^{2,3}

¹ Institute of Aircraft Design (IFB), University of Stuttgart, 70569 Stuttgart, Germany; rudolph@ifb.uni-stuttgart.de

² Institute for Control Engineering of Machine Tools and Manufacturing Units (ISW), University of Stuttgart, 70174 Stuttgart, Germany; karl.kuebler@isw.uni-stuttgart.de (K.K.); oliver.riedel@isw.uni-stuttgart.de (O.R.)

³ Fraunhofer Institute for Industrial Engineering IAO, 70569 Stuttgart, Germany

* Correspondence: schopper@ifb.uni-stuttgart.de

† This paper is an extended version of our paper published in 5th International Conference on System-Integrated Intelligence (SysInt 2020), Bremen, Germany, 11–13 November 2020.

‡ These authors contributed equally to this work.

Abstract: In this paper, a combination of graph-based design and simulation-based engineering (SBE) into a new concept called Executable Integrative Product-Production Model (EIPPM) is elaborated. Today, the first collaborative process in engineering for all mechatronic disciplines is the virtual commissioning phase. The authors see a hitherto untapped potential for the earlier, integrated and iterative use of SBE for the development of production systems (PS). Seamless generation of and exchange between Model-, Software- and Hardware-in-the-Loop simulations is necessary. Feedback from simulation results will go into the design decisions after each iteration. The presented approach combines knowledge of the domain “PSs” together with the knowledge of the corresponding “product” using a so called Graph-based Design Language (GBDL). Its central data model, which represents the entire life cycle of product and PS, results of an automatic translation step in a compiler. Since the execution of the GBDL can be repeated as often as desired with modified boundary conditions (e.g., through feedback), a design of experiment is made possible, whereby unconventional solutions are also considered. The novel concept aims at the following advantages: Consistent linking of all mechatronic disciplines through a data model (graph) from the project start, automatic design cycles exploring multiple variants for optimized product-PS combinations, automatic generation of simulation models starting with the planning phase and feedback from simulation-based optimization back into the data model.



Citation: Schopper, D.; Kübler, K.; Rudolph, S.; Riedel, O. EIPPM—The Executable Integrative Product-Production Model. *Computers* **2021**, *10*, 72. <https://doi.org/10.3390/computers10060072>

Academic Editor: Stefan Bosse

Received: 31 March 2021

Accepted: 20 May 2021

Published: 27 May 2021

Keywords: simulation-based engineering; graph-based design languages; virtual commissioning; product-production model

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

When designing production systems (PS), flexibility with regard to the markets needs (e.g., product variants, production capacity) brings up the need for reconfigurable manufacturing systems [1]. Additionally, the most recent and future challenge for PSs lies within a design process considering energy and overall resource efficiency in the entire life cycle [2]. Requirements from new product variants, which were not foreseen in the first years of operation, and changes in regulations increase the complexity and demand for higher flexibility in the current and future design processes. Due to a lack of knowledge about the effects of changed requirements on today's PSs, the subsequent adaptations are associated with high efforts and costs. The prevailing design process for most PS manufacturers is still based on the sequential waterfall model, first defined by ROYCE et al. [3]. This is in contrast to the idea of more parallelization using methods of the digital factory, cf. [4]. The following two general disadvantages can be identified:

- Solution variants must be excluded at an early stage, as parallel development is not economically feasible.

- Decision-making is discipline-specific (mechanics, electronics, software), sequential and involves compromises, but often ignores cross-disciplinary couplings.

Such deficits and problems have been identified in [5]. The main outcome was that a sequential and “first-time-right” design philosophy prevails in the PS industry. Figure 1 shows phases and activities within the life cycle of a customer-specific PS.

Notable is the *simultaneous engineering* (Definition from [6]: “Parallel processing of different tasks with constant adjustment of the progress made”) of the virtual commissioning (VCOM) phase in parallel to the assembly activity. The pre-drawing of test and optimization steps from the commissioning phase is enabled by a hardware-in-the-loop simulation of the PS, e.g., [7,8]. VCOM allows to increase the quality of the PS right before commissioning and at the same time allows to shorten the remaining time during commissioning by about 75% [9]. VCOM is part of the concept *digital factory*. The goal of a complete digital factory, as described by the Association of German Engineers (VDI), is to combine product and production engineering using digital methods, models and tools (see [10]). Essentially, the integration of various simulation disciplines will help to design, plan and operate product and production in the vision of the digital factory concept [11].

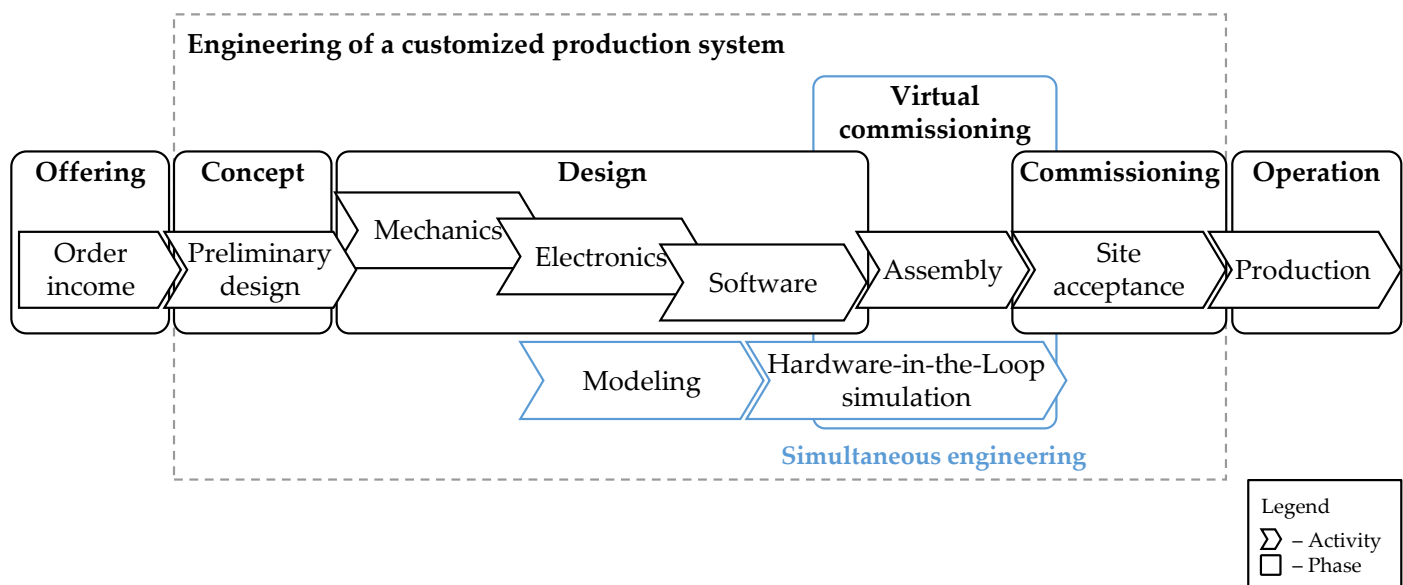


Figure 1. Prevaling engineering approach in the life cycle of production systems.

Derived from the previous work [5], a new Model-Based Systems Engineering (MBSE) approach would be needed to engineer PSs in a more efficient and flexible manner. The motivation behind MBSE in general is to improve productivity [12]. To some extent the complexity of building and reading central data models prevents their adoption in the industry. In research, the concept of digital factories and simulation technology, as an intuitive way to view and execute models, is investigated and approved in numerous ways (e.g., see [13–19]).

In [20] the connection and also the differences between MBSE and a graph-based design approach are shown. Graphs are mathematically formalized models that can hold information from different kinds of engineering models [21]. In the paper on hand the authors elaborate a combination of graph-based design languages and simulation-based engineering to come up with a new model-based engineering concept that copes with the above-mentioned challenges for PS. For this paper, the authors define the term *simulation-based engineering* SBE as an engineering approach for product or production engineering that uses simulation models as a basis for verification and validation purposes throughout the life cycle. Each step between phases of the life cycle needs to pass by a quality gate by passing simulation scenarios.

The content of the paper is organized as follows: First the two central concepts, SBE and graph-based design are introduced. This is followed by an overview of the state of the art, which is critically examined against defined criteria. Afterwards, the overall concept will be presented. This paper features foremost the production design aspects of the concept. For a better understanding, the concept is then exercised by means of an application example. The paper closes with a summary and conclusion.

2. Fundamentals and Literature Review

2.1. Methods for Virtual Commissioning of Production Systems

According to the “rule of ten”, late changes during the life cycle of a system or product result in exponentially increasing costs [22]. Early fault detection and frontloading of changes are therefore an important factor to control cost, time and quality in the making of a system or product. In the development process for PS, VCOM is a state-of-the-art simulation method to “reveal and rectify faults originating from the engineering of an automation system” [23]. To be able to reveal faults from engineering at an early stage of the development process, a mapping of the corresponding PS is needed. Therefore, a simulation model of the PS is simultaneously developed with the real system. Within so-called x-in-the-loop environments the simulation model can be coupled with an instance of the automation control: common configurations are model-, software-, or hardware-in-the-loop (MiL, SiL or HiL) [23]. Due to continuous system modeling for x-in-the-loop environments not only logical but also runtime errors in the control can be detected [4]. Depending on the stage of development of the automation system its modeled behavior (MiL), its real code on an emulated hardware (MiL) or the real code and the real hardware (HiL) is coupled with the simulation model. The use of HiL can be seen as the most-accurate method in comparison to the physical PS as it includes the real hardware control, thus it is used for the following tasks [7]:

- The VCOM of control hardware and software
- Testing of the automation software and the human machine interface
- The optimization of the automation software
- Training of operators for the real production system

The simulation model contains the components, the processes and the communication interface of the PS [23]. Further elements of the simulation model may include special (e.g., physics) behavior, material flow, parts of the human-machine interface. Dependent on the industry division the simulation models are created with a different width of the model and level of detail. E.g. within the simulation model of an entire assembly line the material flow between the assembly modules is very important whereas a robot welding cell has its scope on the kinematics of the robot and collision detection. [23] shows a classification of different model types used in simulation models for VCOM.

Up to now, the disadvantage of using VCOM is the upstream additional effort for model generation for the simulation environment. In the future, simultaneous engineering should allow the simulation model to be created as efficiently and quickly as possible in parallel with the entire development using suitable methods [24,25]. Preliminary work exists on low-effort generation by means of creation from a construction kit with defined cyber-physical modules [26]. Further contributions to low-effort model generation, e.g., by reuse-oriented generation [27], on the basis of an existing construction kit [14,28], on the basis of planning data [28–32], by methodical reuse of partial models [31,33] and by means of model-based mapping of the mechatronic data [15,34] are known. None of the approaches can compensate for the disadvantages that result from the sequential “first-time-right” development approach for PS:

- *Late time of model generation*: Only final information from the individual mechatronic disciplines is included in the simulation models because the intended use is VCOM.
- *Low frequency of model generation*: The simulation is only generated and used for final acceptance, since data maintenance is still very time-consuming.

2.2. Graph-Based Design Languages based on Principles of Systems Engineering and Model-Based Systems Engineering

Graph-based design languages (GBDLs) [35,36] represent an innovative extension of the MBSE approach, see [37], since, unlike MBSE, the central data model is created automatically. This is achieved by mapping concepts (i.e., the *vocabularies*) and assembly knowledge (i.e., the *rules*) into recombinable language modules and operations [36]. In the information representation of a design language, the nodes of a graph serve as abstract placeholders for real objects or processes. Representing the product design, a graph representation (*Design Graph*) is thus processed by machine. This machine processing expands and modifies the graph dynamically at runtime. The abstract product representation allows easy modularization and scalability and enables consistent integration of discipline-specific models (see Figure 2). The generation of discipline-specific models in the target format (DSL) are realized via interfaces. For a more detailed description of how GBDLs work, please refer to [38–43].

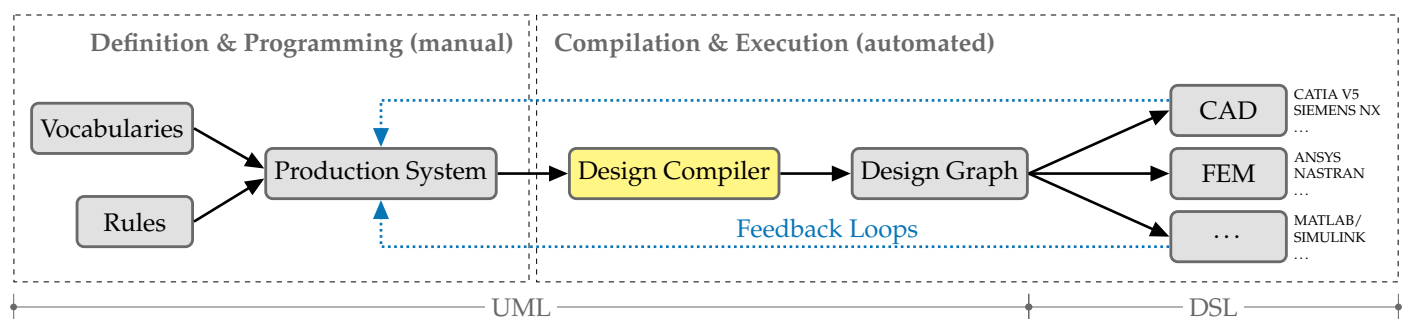


Figure 2. Schematic representation of the information flow in GBDLs [35].

According to the systematic design methodology of Pahl and Beitz [44], the requirements are initially mapped to abstract product functions in the product design, then possible solution principles are elicited and finally the appropriate components are selected. This basic approach is also reflected in the common representation of the development process of mechatronic systems as a V-model (see e.g., [6]). In GBDL, the machine creation and processing of the Design Graph provides the possibility to make the design flow of the product—i.e., the V-model itself—executable [38]. After a design language has been created, the development time for a product variant shrinks to the duration of the translation process and is orders of magnitude below the time of a conventional design loop. The main time expenditure is thus shifted far to the front of the development process, while the many individual design and integration loops can be explored in all variants without any significant additional effort. GBDL is based on the Unified Modeling Language (UML) as the data format. This enables a manufacturer-independent and thus reusable representation of knowledge. The formulation of design and manufacturing knowledge in the form of rules and their linear processing enables reuse and makes the increasing complexity in the product life cycle manageable. GBDL thus provides an innovative and powerful engineering framework that fulfills the requirements set out at the beginning, cf. Section 1 a). Also the central characteristics of the “semantics and syntax”, which are worked out in the BITKOM study, which are “an essential basic condition for [...] interoperability of data storage, data communication and data processing” ([45] pp. 38, l. 1–4) are covered and thus the “integrative development of product, process and production system” demanded in the study ([45] pp. 25, l. 7–8) made possible. In the working group “Design Theory and Similarity Mechanics” under the leadership of the co-author Stephan Rudolph, knowledge-based methods for the design of complex systems based on GBDL have been developed for more than 17 years. In the past, GBDLs were used for the design of aircraft [39] and satellites [40,41,46–48], as well as for the modeling of digital factories for aircraft panels and engine hoods [42,49]. In [50], the authors present in a joint preliminary work how the successful linkage can work in an example. The preliminary work is limited to a rudimentary realization that is

focused on a production engineering module. The development of a complete GBDL for product and suitable production, as well as automatically generated, iterative VCOM, is therefore the task for this application.

GBDLs can map the product design completely digitally. With this type of automated product design, the main effort in terms of time is shifted forward to the creation of the GBDL. In return, the time for design and integration shrinks to the pure running time of the machine translation process. Since the execution of the GBDL with modified boundary conditions (feedback) can be repeated as often as desired, a “scanning” of the design space (DoE) is made possible, whereby also unconventional solutions can be considered. The combination of methods for VCOM and GBDLs thus represents a highly innovative and promising concept for the development of modern PSs. In this work, the software *Design Cockpit 43*[®] is used to execute the GBDL embedded in the EIPPM. For more information about this software please refer to [51].

2.3. Literature Review of Model-Based Engineering for Production Systems

As argued in [5,52,53] the introduction of MBSE in PS engineering is an important step to cope with the ever increasing complexity of PS. Driving motivations are higher product variety and increasing mechatronic interdependencies. However, there is a trade-off between the added value of fully mapped systems and the additional modeling effort [53]. Besides technical improvements the intercommunication of contributors of all disciplines is improved by MBSE methods [52]. The authors are convinced that in the future, model-based, iterative and SBE of PSs will predominate, likewise taking product requirements into account. Due to the sheer volume of publications in the field of design methodology, simulation technology and product-production relations, this work sets focus on contributions from the fields of Model-Based Engineering (MBE) and MBSE for PSs. Both MBE and MBSE are, according to the knowledge of the authors, not used to any significant extent in the PS engineering industry, which is why the topic is still seen as a subject for fundamental research. The authors in [54] identify a high increase in research and publication work on “modeling languages for Industry 4.0 questions” within the last four years of their study (2015–2018). The two main goals of all projects and initiatives can be summarized as (a) using a systems engineering/model-based engineering approach to cope with an ever increasing complexity of PSs, (b) increase the ease of use in creating and using the (central) data models of the PS. The authors see an untapped potential in integrating product requirements with the production requirements, in including variants during the entire development process, in using simulation-based verification and validation in every step of the development and in removing the effort of data model creation. Therefore, the following following requirements are defined, partly formulated in [5], to review state-of-art literature:

- /R1/ Model-based description and coupling of a product and its associated PS (semantic interoperability)
- /R2/ Automatic derivation of solution variants of the PS from the description
- /R3/ Automatic generation of simulation models for the PS
- /R4/ Feedback from simulation-based validation into system design

Table 1 shows the digest of the literature review. As the literature review was carried out with the focus on model-based engineering approaches all of the considered contributions do partly fulfill /R1/. We differentiate between solutions modeling *product*, *production*, *either of them* or *both* (cf. second column in Table 1). Many contributions focus on model-based generation of software code to automate a mechatronic product or PS, e.g., [55–59]. The modeling language *MechatronicUML* cf. [55] is able to model mechatronic systems’ composition and behavior. In the work of POHLMANN [56] *MechatronicUML* is used as a basis and is used to generate simulation models in *Modelica* for model-in-the-loop simulations of the software design. The work is missing a full virtual validation generation and feedback loop /R3/ and /R4/. Besides, *MechatronicUML* itself provides a development process that covers the development of mechatronic systems from formal

specification of requirements until the generation of platform-specific software code. A full representation of product and production is not achieved /R1/. Further work on model-based generation of software code for mechatronic systems can be found from [57] and [59]. Their concepts aim at model-based generation of the software component for mechatronic systems, partly satisfying /R2/, but not for the entire design of a PS. The authors of [57] are proposing a framework that integrates discipline-specific models and couples them, also allowing the inclusion of simulation and verification-specific models. The verification steps do not allow a predefined feedback into the system design, see /R4/. The work of [59] is based on the project *IMoMeSA* [58] funded by the German Federal Ministry for Economic Affairs and Energy (BMWi). *IMoMeSA* is the successor of the project *AutoVIBN*, cf. [60]. The outcome for *AutoVIBN* was a functional modeling technique for the modeling and generation of VCOM models. The succeeding project *IMoMeSA* extended the modeling technique from *AutoVIBN* by integrating requirements and *failure mode and effects analysis*. The latter allows the modeling of erroneous behavior of the PS. With the extended model the generation of software code, discipline-specific models like CAD (Computer-Aided Design) and FEM (Finite-Element-Method) is realized. The combined contribution from *AutoVIBN* and *IMoMeSA* is not able to model both, product and production with their interdependencies, cf. /R1/, and to include variants in the data model, cf. /R2/.

In product development ALBERS et al. contributed in several works [61–64] to enable MBSE approaches and foster re-use of information in multiple ways. Between product generations, knowledge can be transferred using product profiles linked to product functions [64]. For the continuous validation of products, the authors aim at the re-use of knowledge of validation environments by using elements of a reference system [65]. The authors do not go into detail about the creation of Simulation-based validation environments /R3/.

The work of BURSAC [66] is a contribution towards a more efficient development of products using construction kits. He combines construction kit-based engineering with the concept of Meta Object Facility™ (cf. [67]) and the Integrated Product Engineering Model (cf. [63]). Its goal is to promote an iterative product development process. The approach relies heavily on the existence of reference products to create reference models for the construction kit. VCOM does not take place /R3/ and the production matching the product /R1/ is not considered.

Within the project *KitkAdd* [68], JACOB et al. developed a model-based method to couple product characteristics with production technologies [69,70]. From the models, technology chains can be derived, compared and iteratively improved. Improved parameters can then be followed up to reduce production costs. The approach does not use simulation-based validation /R3/ and is limited to existing solutions of manufacturing units /R2/, not considering all disciplines of the design process.

The systematics resulting from the project *mecPro²* [71] describes a holistic data model based on SysML (see <https://www.omg.sysml.org> (accessed on 25 March 2021)). Three different dimensions are declared: variability, detailing, concretization. In addition, four levels (context, function, solution principle, component) are identified. The result is an extensive model-based coupling of product design processes, methods and IT. For the administration of data models the authors developed an integration with existing Product Lifecycle Management systems. The data model of *mecPro²* is only descriptive and not executable /R2/, as the SysML language lacks a semantic. For each addition to the model its semantic must be extended. The *mecPro²* approach uses models for simulation modeling and linking with native simulation files to enable a simulation-based verification /R3/. SBE is not fully enabled as native simulation files are needed and no X-in-the-Loop environments can be generated.

The approach of OESTERSÖTEBIER [72] enables a semantic linking of knowledge for the modeling of complex mechatronic systems. However, Oestersötebier derives from the a state of the art analysis, that an automation of the design process does not seem reasonable /R2/ and limits his contribution to the support of the development by: efficient reuse, support of information exchange, both based on a model-based design.

Representative for the research in the field of integrated product and production development at SAARLAND UNIVERSITY is the work of VIELHABER et al. The authors in [73] e.g., specify an integrated selection process that considers production process, material and product in combination. The combinations are generated with the help of a morphological box and subsequently evaluated under consideration of economic, ecological and technical aspects and visually represented in a three-dimensional coordinate system. The approach represents a pure process description which cannot be used for an automated generation of artifacts, not fulfilling /R2/ and /R3/. Other influences, such as the shape or physical effects (e.g., mechanical loads) are not taken into account /R1/.

A contribution aiming at the continuous generation of simulation models throughout the development process /R3/ is the project WIEMOD, cf. [74–77]. The main goal for the project was an earlier usage of simulation and a reduced effort in creating the simulation models. VERL et al. [74] reduce the creation effort by adding simulation models with mechatronic units in a construction kit, where every component instantiates its simulation model effortlessly. Not considered within the solution are product requirements that influence the PS and the creation of the PS variants /R2/.

Table 1. Evaluation of the reviewed literature according to the criteria /R1/–/R4/.

Reference	/R1/	/R2/	/R3/	/R4/
MechatronicUML and Pohlmann [55,56]	○ either of them	●	●	○
Alvarez Cabrera et al. [57]	○ production	●	○	●
Project AutoVIBN [60]	○ production	○	○	○
Project IMoMeSA and Hackenberg [58,59]	○ either of them	●	○	○
Project KitkAdd, Jacob et al. [68–70]	● both	●	○	●
Project mecPro ² [71]	● both	○	●	●
Albers et al. [61–63]	○ product	○	○	○
Bursac [66]	○ product	●	○	○
Oestersötebier [72]	○ product	○	○	●
Stoffels et al. [73]	● both	○	○	○
Project WieMod and Voß [74–77]	○ production	○	●	●

○ not fulfilled; ● partly fulfilled; ● fully fulfilled

There is a variety of standardized and quasi-standardized modeling languages to represent PSs in the context of the digital factory. In [78] SysML, MARTE (see <https://www.omg.org/omgmarte/> (accessed on 25 March 2021)), CMSD (see <https://www.sisostds.org> (accessed on 25 March 2021)), AutomationML (see <https://www.automationml.org> (accessed on 25 March 2021)), Modelica (see <https://www.modelica.org> (accessed on 25 March 2021)), Eclipse Ditto and Vorto (see <https://www.eclipse.org/ditto/> and <https://www.eclipse.org/vorto/> (accessed on 25 March 2021)), VDI/ VDE 3682 [79] and Project MAYA (see <http://www.maya-euproject.com> (accessed on 25 March 2021)) are examined. We conclude from the work, that (a) none of the modeling languages is able to represent a PS as a whole system, (b) only some of the modeling languages are suitable for model-to-text transformations to generate simulation models, (c) None of the languages allow a coupling of product and production features.

2.4. Interim Conclusions

The authors did not identify any literature dealing with all four requirements simultaneously. Most contributions focus on a specific way to facilitate the creation and the usage of a (central) data model for MBE/MBSE purposes. The state of the art however shows that the model-based description of either products or PSs has been researched widely. The authors therefore do not see their contribution primarily in the creation of an additional description language, but in the way in which the description language is used. All the referred contributions on MBE and MBSE for PSs fall short on the idea that the data model itself is created manually versus generated in an automated way. This paper focuses

on capturing the knowledge of involved disciplines to be able to automate the creation of the central data model. On top of a facilitated model creation, multi-domain discussions are enabled by simulation-based verification and validation.

The use of GBDL and SBE fundamentally changes the distribution of roles in the development process. In the future, design models will be jointly defined by system theoreticians together with engineers from mechanics, electronics and software. The solution variants will be created automatically and without the intervention of an engineer. Only the evaluation and selection of a solution variant is still the responsibility of the engineer. The decision making process will be supported virtually throughout. In the virtual representation, the authors see the optimal form of knowledge representation for all stakeholders involved. In current MBE approaches the representation is usually neglected [54]. The approach presented in this paper is entitled *EIPPM—The Executable Integrative Product-Production Model*. In order to achieve a real benefit, the authors define the following goals, derived from the requirements /R1/-/R4/, for the EIPPM:

- Formal storage of knowledge in the form of class diagrams and rules for simple know-how reuse
- Full semantic interoperability in the modeling of the product and the associated PS
- Fully automatic derivation of solution variants
- Fully automatic generation of simulation models for continuous virtual verification
- Feedback system of simulation-based validation and optimization back into system design model

Further positive side-effects, partly formulated in [5], are: First, a reduction of the effort for simulation model creation. Second, a consistent connection of the mechanics, electronics, software and simulation disciplines throughout the engineering. Third, generation of simulation models starting with the planning phase.

In the following section, the EIPPM is introduced. The interaction of product and production is achieved via the EIPPM. In addition, the dynamic data model of EIPPM is described and how the simulation models are integrated there.

3. The Executable Integrative Product-Production Model (EIPPM)

The innovation of the EIPPM presented in this paper stems from the fact that all models and artifacts needed during the development process of a PS—taking into account also the interconnection of product and PS—are generated *fully automatically*. This total design automation involves applying modern modeling principles, procedures and data structures that originate in computer science. GBDLs (see Section 2.2) provide all these prerequisites and are therefore used as a basis here. In the following, the fundamentals of the methodology underlying the EIPPM will be described.

A general and abstract approach to product design is offered by the systematic design methodology according to PAHL and BEITZ [44]. Thereby the requirements are initially mapped to abstract product functions, then possible solution principles are determined and finally the suitable components are selected. The components are then assembled into subassemblies, which in turn are finally assembled into the product. This basic approach is also reflected in the common representation of the development process of mechatronic systems as a *V-model* (see e.g., [80]). In GBDLs, the automatic creation and processing of the Design Graph makes it possible to automate the steps of the V-model [38]. The unique possibility of describing interrelationships of different domains via linked class diagrams (*domain ontologies* respectively *vocabulary*) enables a continuous and consistent description of all tangible and intangible elements of the design process. This means that data of conceptually separate domains does not have to be linked manually at a later time (as in most of the approaches presented in the state of the art), but the interlinking is already contained in the “structure” of the modeled class diagrams. Semantic interoperability is thus an inherent feature of the graph-based methodology. Once the required classes have been defined, instances of these classes can be created and combined with each other in rules (in accordance with the dependencies defined in the class diagram). A compiler

assembles possible variants of the system according to the defined rules. This process fulfills the condition formulated in /R2/ (see Section 2.3). Since the class diagram restricts the possible combinations, it represents an envelope of all possible designs—the *design space*. The rules and the translation process in the compiler then serve to “scan” the design space and thus to find the best possible solution as a unique connected instantiation of the vocabulary. The design space defined by the class diagrams is always larger than the meaningful one, since not every possible instance of a class is technically valid: For example a mass attribute or a length may never have a negative value. There are two fundamentally different ways of dealing with this problem. Either the rules are formulated in a way that prevents a “degenerate” design or the “wrong” designs are sorted out in a subsequent step.

After a GBDL has been created, the development time for a product variant shrinks to the duration of the compilation process and is orders of magnitude faster than conventional design loops. The main time effort is now spent on the creation of the GBDL, while the design and integration loops can be explored in all variants without significant additional effort. Figure 3 illustrates how in the EIPPM product and production design are combined into *one single* design cycle and how the complex relationships can be ensured to be consistent. In this way, requirement /R1/ (see Section 2.3) is fulfilled.

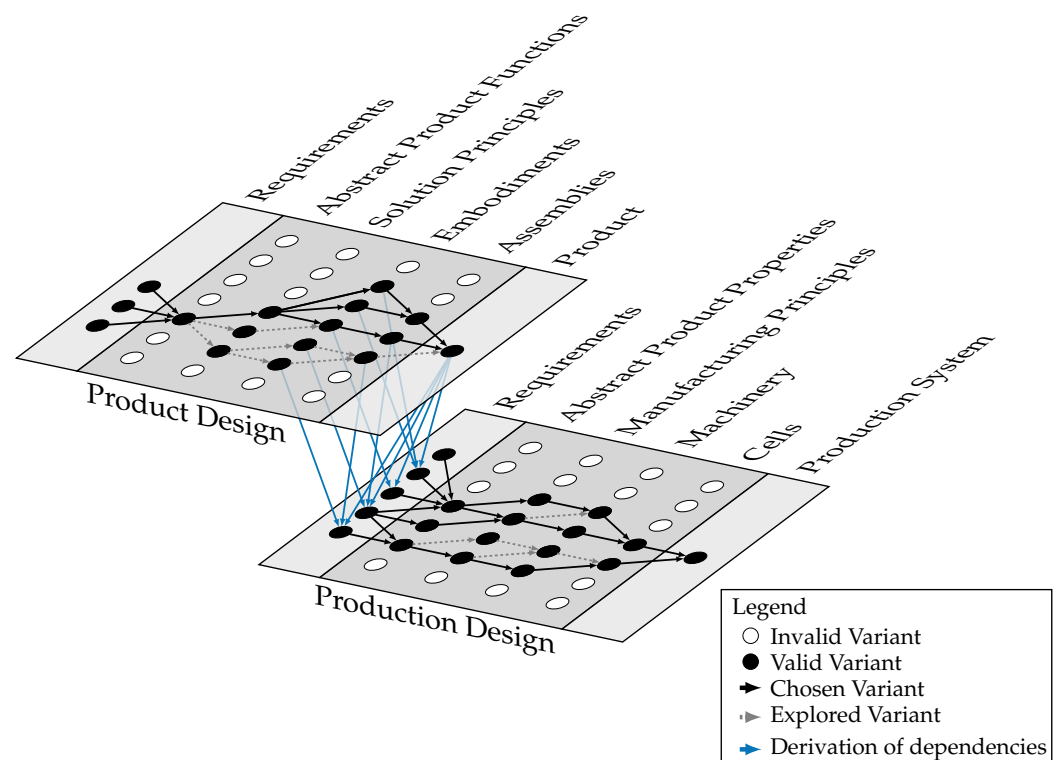


Figure 3. Dependence of the product and production design in the EIPPM.

The product design and the production design are shown on two different levels. This is to better illustrate the time sequence, where production planning usually follows product design. Depending on the product being produced, a different PS is required. The structure within both design levels is based on the aforementioned systematic design methodology [44]. Here it is important to emphasize the combination of a well-researched design methodology (Pahl/Beitz) with the innovative possibilities of GBDLs. From a practical point of view, this means that for each level of abstraction—requirements, abstract product functions, solution principles, embodiment, assemblies and product—the required building blocks must first be formulated in the form of a class diagram. This modeling process applies to both product- and production system design. In addition, associations and inheritance are used to define which module (class) can be combined with which other

module. For example, the class diagram defines which type of requirement can be implemented with which abstract product function. It is also determined which abstract product function can be realized by which solution principle and so on. The actual instantiation and combination instruction is defined in rules. The compiler will then generate a valid variant for the boundary conditions applied. In this context, “valid” means that the initially defined boundary conditions for the product and the PS can be fulfilled. No statement is made about how well these are met compared to another variant. This must be decided in a downstream evaluation step. In Figure 3 this is indicated in by the filled black circles and the arrows between the stages. White circles are to show that an instance is not valid under the conditions defined by the previous step. The dotted arrows are intended to indicate that the compiler has examined several instantiations. This varying instantiation process is ultimately responsible for the generation of variants and the presence of multiple variants in a single design graph (“150% model”). The blue arrows (between the levels) show mappings between the product and the PS. This means that the specifications for the product can be interpreted as requirements for the production design. In this way, product and production are naturally linked. Semantic interoperability makes it possible to resolve this without further intervention. All that is needed is to formulate appropriate rules for the production design. Building on the decisions already made for the product, the further design of the production can partially be inferred from this. Of course, there are also requirements for the production design independent of the product. This is symbolized in Figure 3 by a requirement not linked to the product design. Finally, the methodology is validated by means of automated simulations.

The integrated generation of simulation models needs to meet certain requirements. As the authors focus on simulation models for VCOM of PS, certain simulation disciplines are of special interest: logical behavior, kinematic behavior, physical behavior, process behavior and material flow. The aspects of all those simulation (model) types must therefore be integrated into the GBDL. The modeling with class diagrams and rules will be able to extend the GBDL with a meta-simulation model to cope with this requirement. Standard VCOM simulation models consist of one behavioral model and a second visualization model. The principal part of information for both models will be handled in the modeling of the design space: The current creation of simulation models uses the same source of information (Mechanics CAD, Electronics CAD, functional specifications, manuals and compendiums, communication specifications and many more) that the GBDL comprises. When advancing with the modeling of the design space a validation of the modeled information is important. In this sense, the requirement /R3/ is fulfilled. From the evolving simulation models it will be clear to see whether errors in form of missing or wrong classes, attributes and constraints are present. The evolution of the system will also be represented in the simulation system’s characteristics. Ranging from MiL simulation with a rather abstract simulation model and a model-based description of the control code to MiL where control code is tested on an emulated control to HiL simulation using the control code and the real hardware control.

The simulation model is to be used from two aspects. First, to verify the design space spanned by vocabulary and rules of the GBDL and the derived Design Graph in the sense of the PS (e.g., scientific laws, signal processing or error behavior). Second, in the sense of requirements from the product and customer (including *cycle time*, *cost*, *scrap rates*, etc.). Both test scenarios lead to feedback into the design of the GBDL thus fulfilling requirement /R4/ (see Section 2.3). How further variants of the product-production system can be evaluated in the EIPPM is explained in the following section.

Iterative Generation of the Product-Production Model, the Simulation Models and the Assessment

The aim of the EIPPM is to transfer the flexibility of product design to the development of production design. This is to be achieved by creating a large number of variants. The prevailing “first-time-right” methodology must be subjected to a paradigm shift [31].

An automated, simulation-based and iterative development philosophy across all disciplines of product-production system development is indispensable for this. The authors see the following four main advantages in such an approach:

1. Automated model-based system development for a complete and consistent description of the complex “product-production” system
2. Iterative approach by automated expression and analysis of variants of the PS on VCOM simulation models
3. Supporting methods for the optimization of the variants or for the optimal selection of a variant
4. Simulative identification of weak points in the generated product-production system and transfer into improvement suggestions for the automated design process

To get a better understanding of the EIIPPM, the automated run through of a design cycle is schematically shown in Figure 4. As already explained in Figure 3, the phases requirements, abstract product properties, manufacturing principles, machines, cells and PS are traversed. The data model was imaginary “sliced” in Figure 4 at several characteristic stages of the production design. The structure of the slices can be read off at any time with the overall model increasing in size as time goes on, represented by “annual rings”.

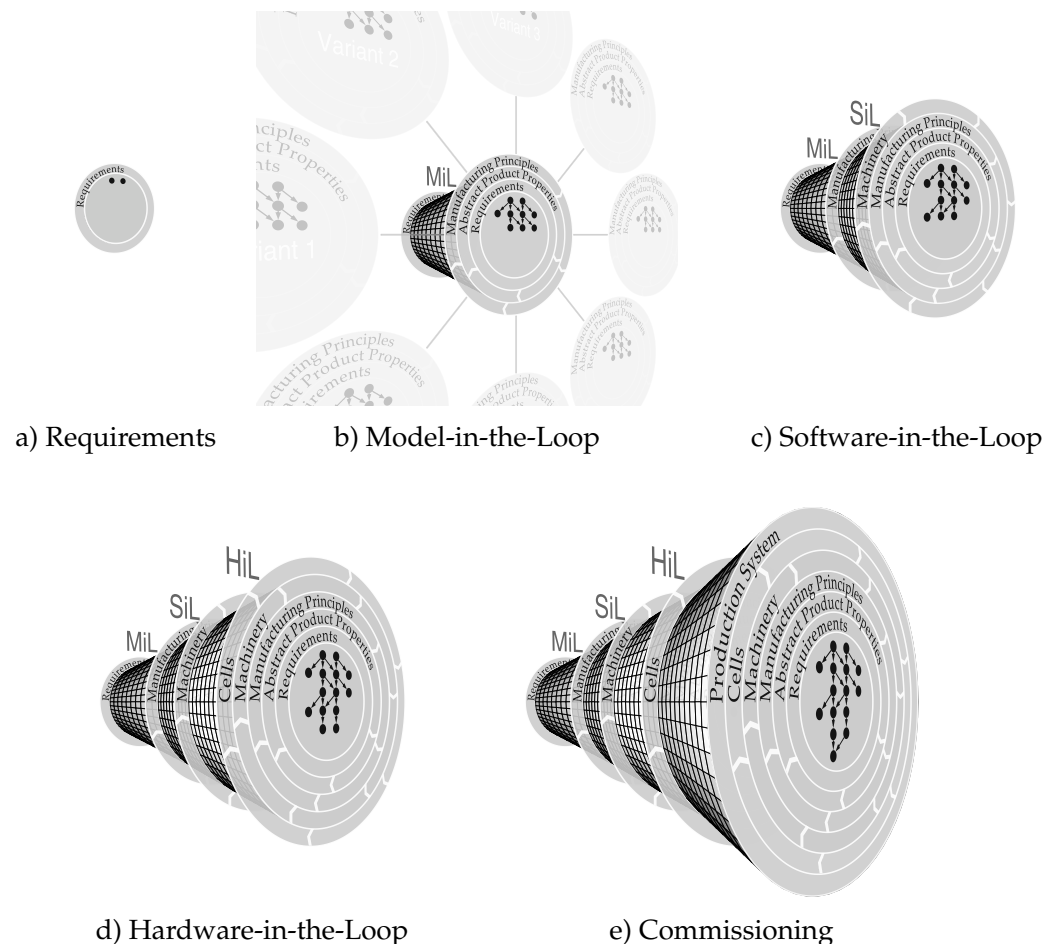


Figure 4. Dynamic enrichment of the the EIIPPM.

Each individual design phase is a closed iteration cycle in itself, which is influenced by the layer directly underneath. For example, the *Abstract Product Properties* can be generated from the *Requirements* by iterating over all possible product properties. Once the *Abstract Product Properties* have been defined, it is possible to move to the next layer—in Figure 4b) the *Manufacturing Principles*—to initiate the next iteration loop. For the VCOM, as a

central element of the EIPPM, three time slices are of particular interest: In addition to the final HiL simulation (Figure 4d)), these are the MiL- (Figure 4c)) and MiL (Figure 4b)) simulations performed at earlier points in time according to [23]. The MiL phase is at a very early stage of the design cycle. The data model is at a rough, descriptive level and includes the Requirements, the Abstract Product Functions and the Manufacturing Principles. From this data, the MiL simulation model can be generated automatically and used to verify the overall data model created to this point. If the model is not executable or only executable to a limited extent, the current run is aborted and a new run with changed boundary conditions is triggered. The generation of MiL simulations is similar, but with a more detailed data model. At the time of HiL simulation, the data model is very detailed (cell level) and includes machine code executable on the hardware (real control, simulated PS). This model can also be generated fully automatically via an interface from the EIPPM. When all iteration loops and associated simulative tests have been successfully run using VCOM, a PS is completely designed. Since the design problem is usually not completely and unambiguously defined, multiple solutions can be expected. This is shown in Figure 4b) by the orbiting slices, each of which represents an individual PS, that still has to go through the further process of verification. The selected variant can then be built.

While Figure 4 explains the design process with the EIPPM, Figure 5 shows the correlation of artifacts in the approach. The levels L0 to L3 are inspired by the levels according to the Meta Object Facility™ (MOF) by the Object Management Group [67,81]. As described in Section 2.2 the GBDL approach uses UML and thus MOF as the modeling framework, cf. L3. When using the EIPPM, the system design model, on L2, must be defined first using the modeling framework. It contains all permissible variants on type level of the system (the so-called 150% model, cf. [82]). With the instantiation in the virtual world on L1, in the form of the system instance model, a permissible variant on type level is generated from the 150% model. The automation of the instantiation is made possible by applying the set of rules on the ontologies, on L2. The information contained in the system instance model can be transferred to a discipline via interfaces, e.g., generate a simulation model. The simulation model is a model of reality, in this case of a PS. Another interface can generate e.g., CAD data from the system instance model (cf. [83]). L0 represents the objects of the real world—product and PS. The challenge is the feasibility of an automatic generation of meaningful and comprehensive variants for the simulation-based solution selection. The mapping of facts, components and couplings of a product-production relation is non trivial and requires the cooperation of different domain experts.

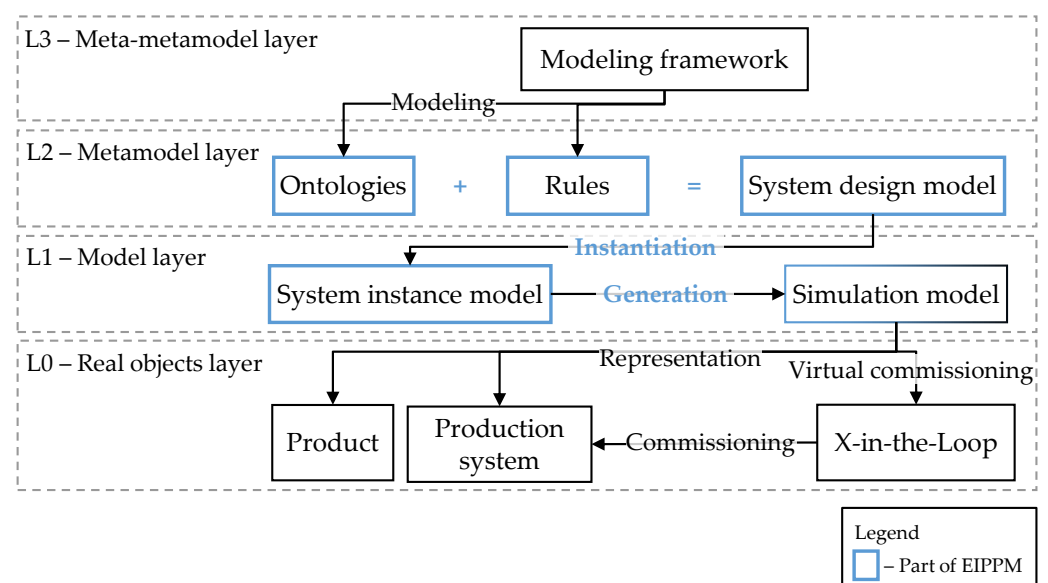


Figure 5. Target architecture of the models and layers involved in the EIPPM development process; layers according to Meta Object Facility™ [67].

4. Application Example

In order to illustrate how the EIPPM works, the modeling of an assembly line PS will be described in this section at a basic level. The example assembly line is an Industrial Mechatronic System (IMS) by Lucas-Nülle[®], installed for educational purposes at the Institute for Control Engineering of Machine Tools and Manufacturing Units at the University of Stuttgart. The IMS is built from real industrial actuators, sensors, communication system and control and is capable of building eight product variants from three different parts. All parts and product variants of the example are shown in Figure 6.

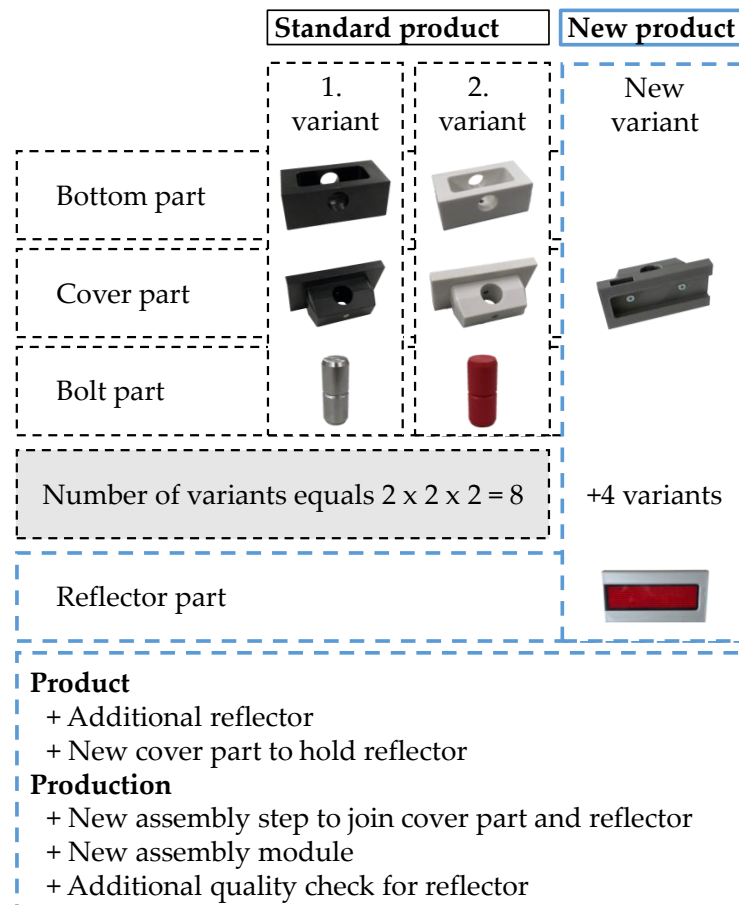


Figure 6. Left: Standard product variants available for assembly on the Lucas-Nülle[®] IMS; Right: New variant with additional functionality adds new requirements to product and production.

To illustrate the interdependencies of product and production a new cover part is introduced with a new functionality of a reflector part, see right hand side of Figure 6. Listed below the new part in the figure are the modifications necessary for product and production.

The IMS assembly line has a ring-shaped topology. Eight straight and two 180°—curved conveyor belts move workpiece carriers around in order to reach the necessary assembly station. Assembly stations consist of the conveyor belt and an assemble module of which three types are used to provide the parts *bottom*, *cover* and *bolt*. A fourth type of station includes a Handling module to remove finished products from the workpiece carrier and the assembly line. Figure 7 shows the entire PS with its seven stations and the conveyor belts.

For each valid product one bottom part (black or white color), one cover part (black or white color) and one bolt part (red or silver color) is selected and sent as an order to the IMS. The control's task is then to assemble the product as defined in the order. The assembly process begins by moving a workpiece carrier via a conveyor belt into the

first station with the appropriate *Separate* module for the bottom part. On the base plate of the *Separate* module are two profile rods that serve as part storage. Inside of it, the bottom parts, are stacked to a tower. By a pneumatically actuated rocker arm, the stacked parts are separated from each other and dispensed individually onto the workpiece carrier. Next, the workpiece carrier is transported to the appropriate *Assemble* module to include the cover part into the product. Except for a diverging geometric of the rocker arm the module's functionality is the same as the one of the *Separate* module. The stored cover part is separated through a rocker arm and dropped onto the bottom part. Then, the workpiece carrier is moved to the appropriate *Finish* module, where a pneumatic cylinder inserts the bolt part by pushing it through the holes in bottom and cover part. Like this, the final product is fastened. Finally, the workpiece carrier is moved beneath the swivel arm of the *Handling* module. The arm is equipped with a vacuum gripper and can lift up the product through a cavity in the base plate from the pickup position, rotate it by 90° and place it down on the deposit.

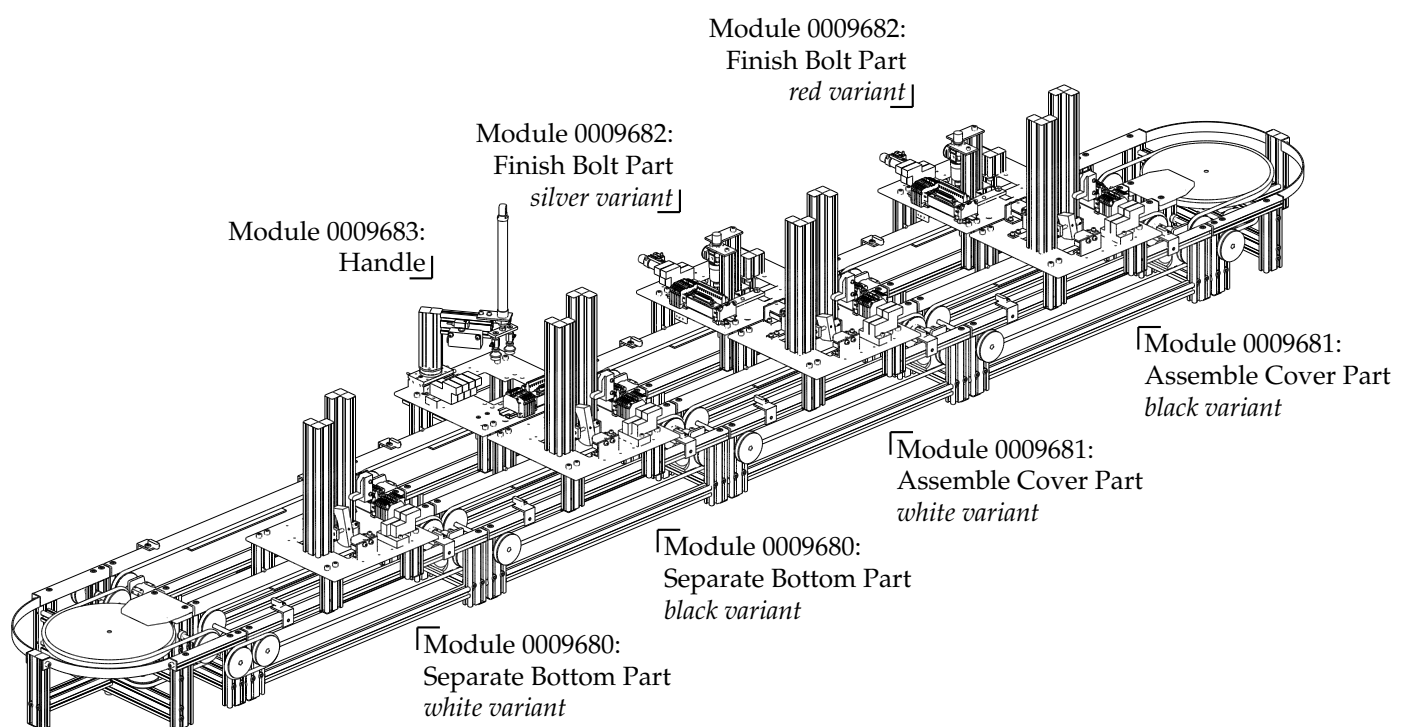


Figure 7. The seven-module example production system IMS by Lucas-Nülle® installed at ISW.

4.1. Modeling of the Class Diagram

The modeling process of the EIPPM starts with the definition of the class diagram. Figure 8 shows a simple example class diagram for the IMS to be described in this section (Note: If you are reading this article on screen, you will be able to zoom into the class diagram as it is a vector graphic file). The depicted class diagram does not claim completeness and is only intended to give an impression of how the EIPPM can be implemented. An industrially applicable class diagram with all details and dependencies would go beyond the scope of this paper and would not be useful for illustrating the process.

A typical first approach to modeling the class diagram is a mental decomposition of the system. The decomposition can refer to different properties, for example geometrical, mechanical, functional or discipline-specific. In this example, we started with a decomposition based on the mechanical components. The classes in the third line of the class diagram, e.g., *Frame*, *Base_Plate* and *Stand* are examples of such component-oriented classes. In addition to the mechanical functionality of the modules, there are still further

dependencies that must be taken into account in the modeling. The dimensions and spatial positioning, materials and simulation fragments can be mentioned here.

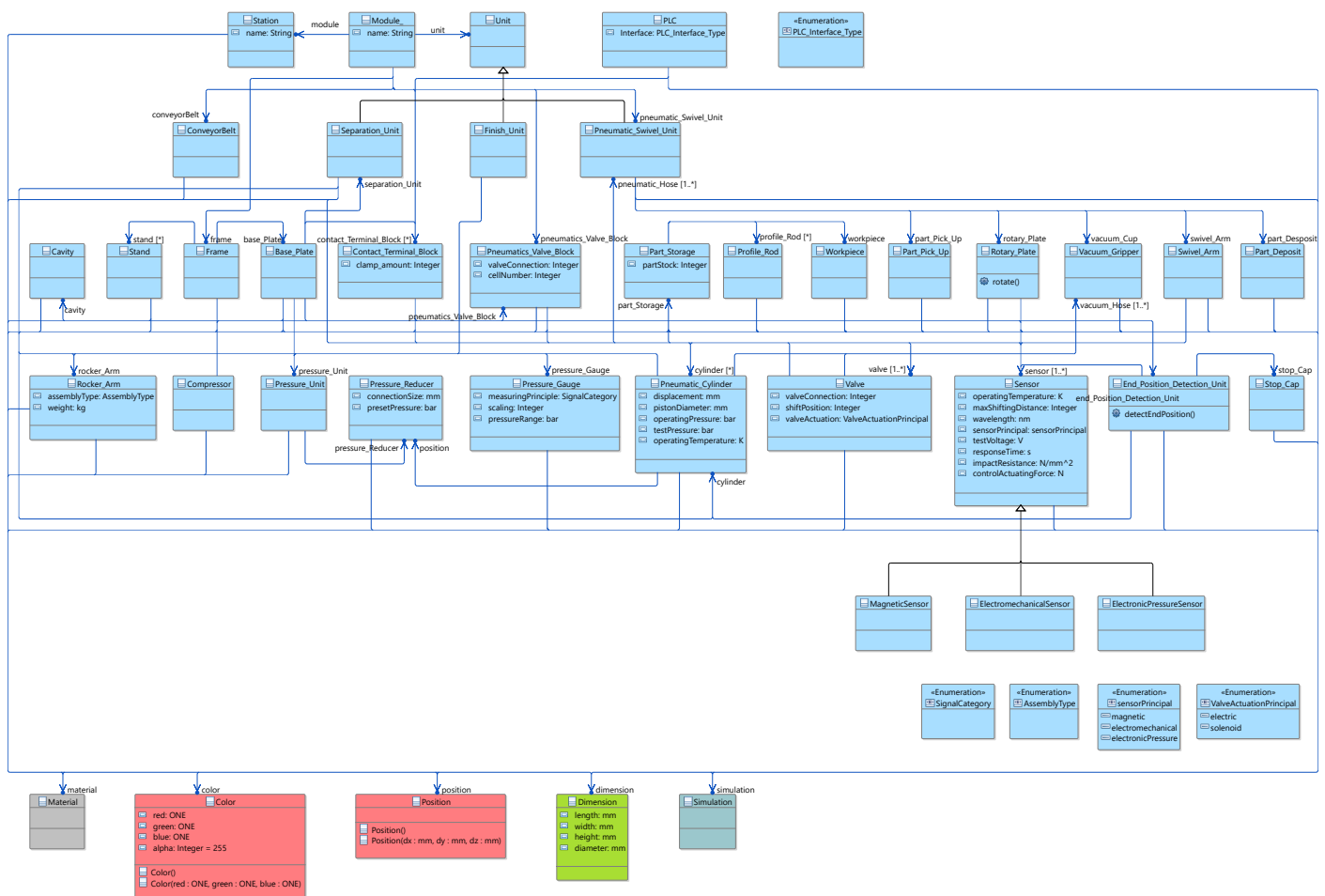


Figure 8. Example class diagram of the PS from Section 4 in the EIIPM.

Class diagrams can be nested, that means that not all information and dependencies have to be included into a single class diagram. An import mechanism allows classes from other projects to be accessed. This enables a sensible encapsulation of design knowledge. A common approach here is to first model independent class diagrams for the domains involved and then map the dependencies of these classes in a separate integration class diagram. In the example class diagram above, the different coloring of the classes `Material`, `Color`, `Position`, `Dimension` and `Simulation` at the very bottom indicates that these classes were imported from other class diagrams.

Since the modeling of the individual modules of the example PS is very similar from an abstract point of view, only the Handling module is described in more detail in the following. In design languages, there is a rule of thumb that the nouns used in a domain are the candidates for the classes. With this in mind, the Handling module is shown again in detail in Figure 9, where the relevant components are named. These names are the basis for the transfer to the class diagram. Figure 10 shows the relevant section of the overall class diagram (see Figure 8) in an enlarged view.

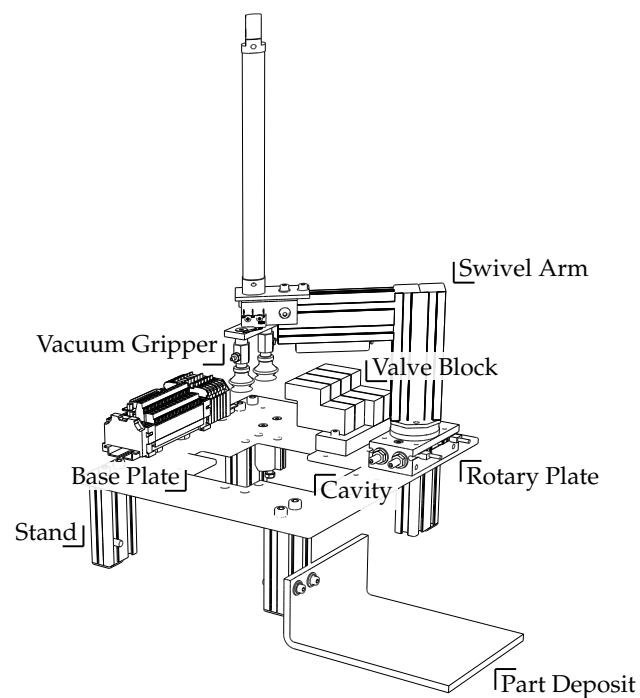


Figure 9. Handling module of the IMS.

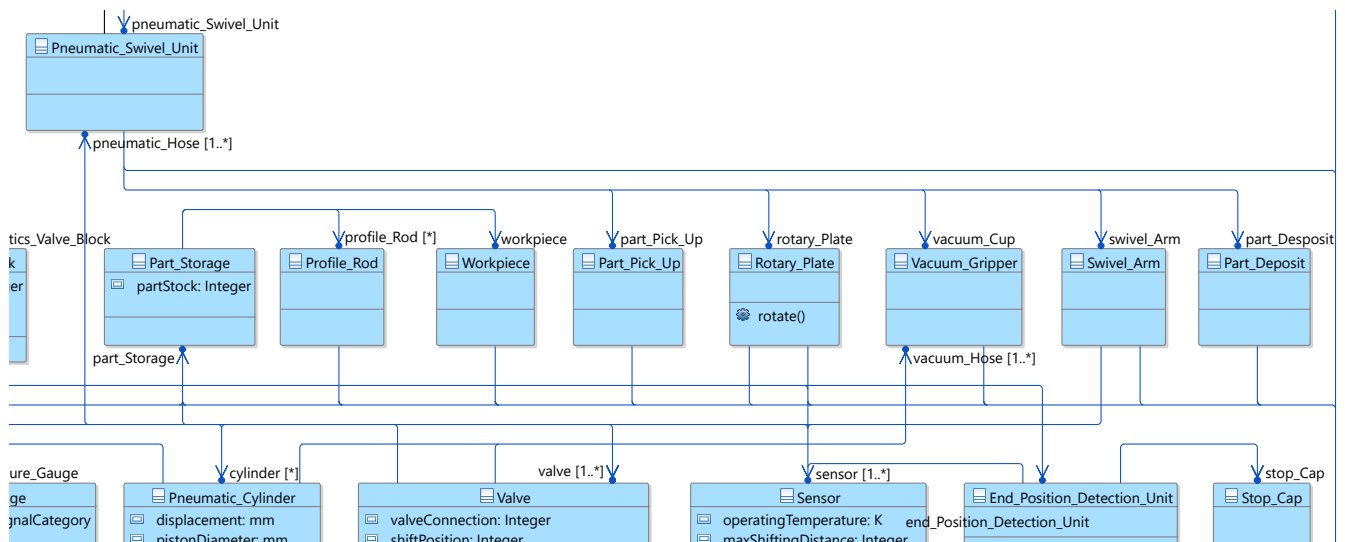


Figure 10. Enlarged view of the class diagram from Figure 8 with focus on the area belonging to the Handling module.

The entire module is represented by the `Pneumatic_Swivel_Unit` class, which in turn inherits from the more abstract class `Module` (cf. Figure 8). This class has dependencies (in the terminology of the UML so called associations, read: *has-a*) to its associated classes. These are for example the classes `Part_Pick_Up`, `Rotary_Plate`, `Vacuum_Gripper`, `Swivel_Arm` and `Part_Deposit`, where the representation being apparent from their names. The dependencies beyond the system boundaries can also be identified. All mentioned classes have an associated `Simulation`, `Dimension` and `Position` class. Furthermore, the classes `Rotary_Plate` and `Swivel_Arm` have a *sensor*-association. This association is provided with a multiplicity (`[1..*]`), which means that there can potentially be several sensors for one component. There are also associations that point to classes in this section view, e.g., the *vacuum_Hose*-Association. This association represents vacuum lines that connect the components of the module.

4.2. Modeling of the Activity Diagram and the Rules

There is next the need to instantiate concrete objects and links from the abstractly formulated classes and associations. This action requires the definition of rules and activities. Just like class diagrams, activities can be nested and imported. Figure 11 shows the overall activity diagram. The subprograms *separate*, *assemble*, *treat* and *handle* can be recognized, as well as the two graphical rules *AddPLC* and *ConnectPLC*.

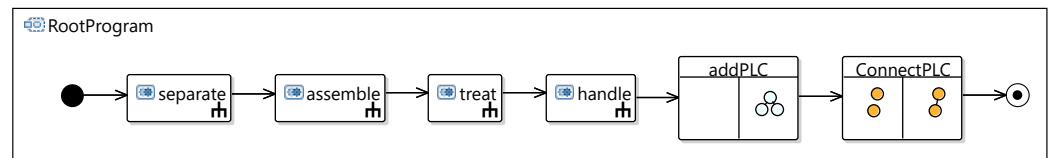


Figure 11. Activity diagram of the overall IMS.

At this point again we want to concentrate the description of the activities and rules on the Handling module in order to avoid repetitions. The modeled activity diagram of the Handling module consists of 12 graphical rules and is shown in Figure 12.

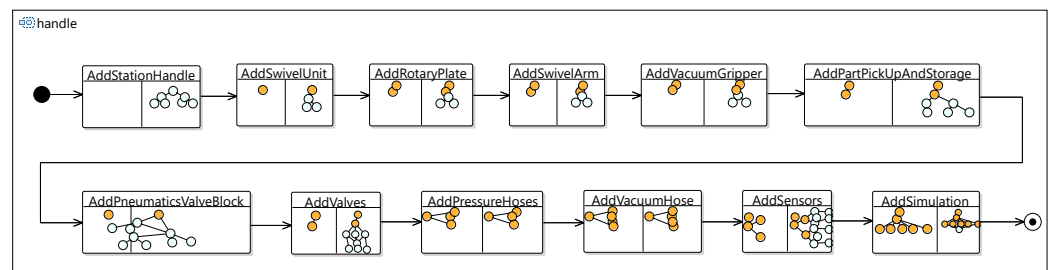


Figure 12. Activity diagram of the Handling module of the IMS.

In the activity diagram the order in which the compiler should translate the rules is defined. In above case, the handling module and the associated station are first created in rule “AddStationHandle”. Then, in rule “AddSwivelUnit”, the *Pneumatic_Swivel_Unit* is instantiated and linked to the module. Next, the *Rotary_Plate*, the *Swivel_Arm*, the *Vacuum_Gripper* as well as the *Part_Pick_Up* and the *Part_Storage* objects are created and linked in the corresponding rules. After that the *Pneumatics_Valve_Block*, the *Valves* and the *Pressure* and *Vacuum-Line* Links are instantiated. Finally, the *Sensors* and the *Simulation* instances are added to the data model (Design Graph).

Graphical rules all have the same structure. There is a Left- (LHS) and a Right-Hand Side (RHS). The graph pattern defined on the LHS is searched for on the Design Graph created so far. If it is found, the action described on the RHS is executed. Figure 13 shows the in-depth structure of the graphical rule “AddVacuumGripper”.

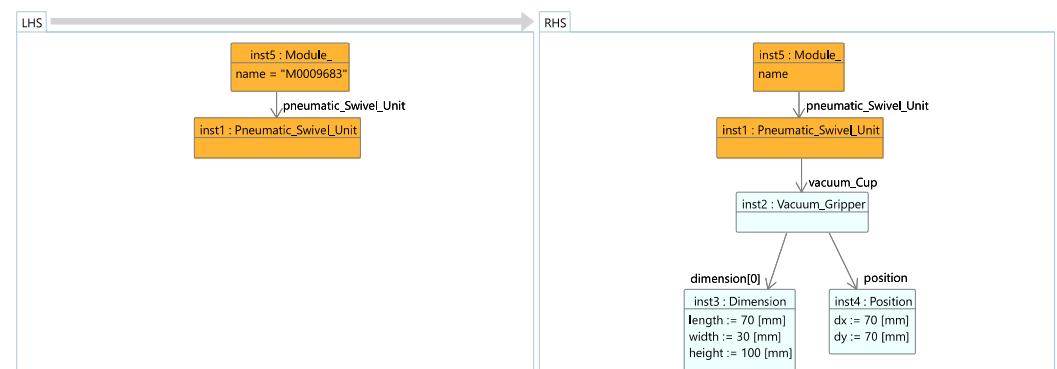


Figure 13. Rule “AddVacuumGripper” of the Handling module of the IMS.

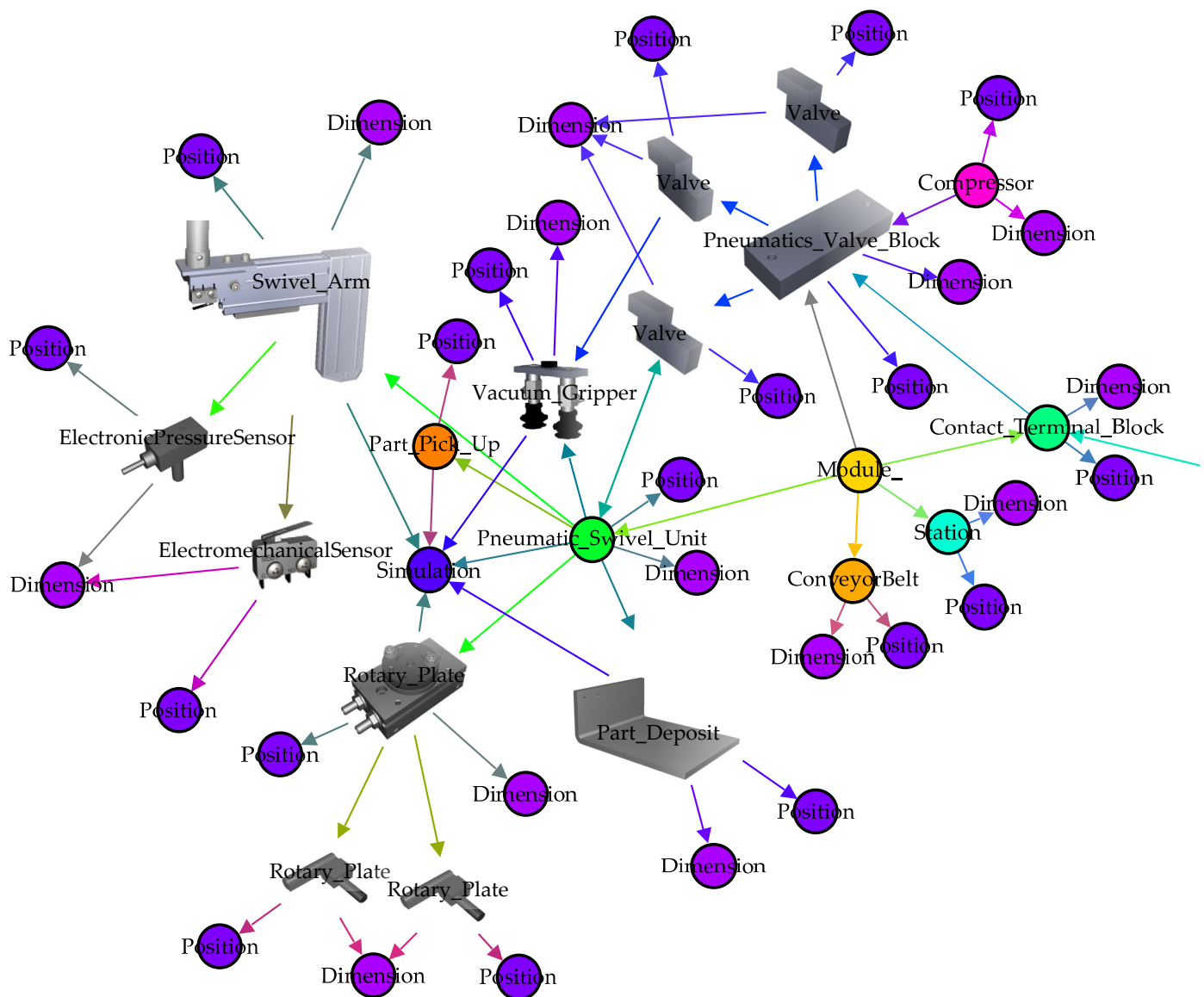


Figure 15. Design Graph of the Handling module after the execution of the EIPPM.

In the figure, for all nodes that have a geometric representation (shape), the abstract circles of the default graph view have been replaced with images of the represented component.

4.4. Generation of the Simulation Models

Apart from the concept given in Section 3 for the derivation of simulation models—which would require the formulation of numerous rules that would go beyond the scope of this educational example—a simplified way to couple a simulation library with the GBDL data model was chosen here: For a straight forward way of generating simulation models the graph from the last section, see Figure 14, is extended by a placeholder object *Simulation*. Each module is linked with its own simulation placeholder, that represents an object from the simulation library. In this case the composition of the simulation model will be derived from elements contained in a library. A generator (model-to-text) will then be able to transform the composition into the executable simulation model. The elements in the library are predefined solutions in a specific simulation tool and can be parameterized and connected together. For the application example the granularity ends at the module level which corresponds to the *Cells* level in the EIPPM. This leaves the observer of the class diagram with a black box representation of the simulation model, equivalent to

the representation in the simulation tool, see Figure 16. The simulation model is only shown for the Handling module. For the other modules, however, the procedure is completely analogous.

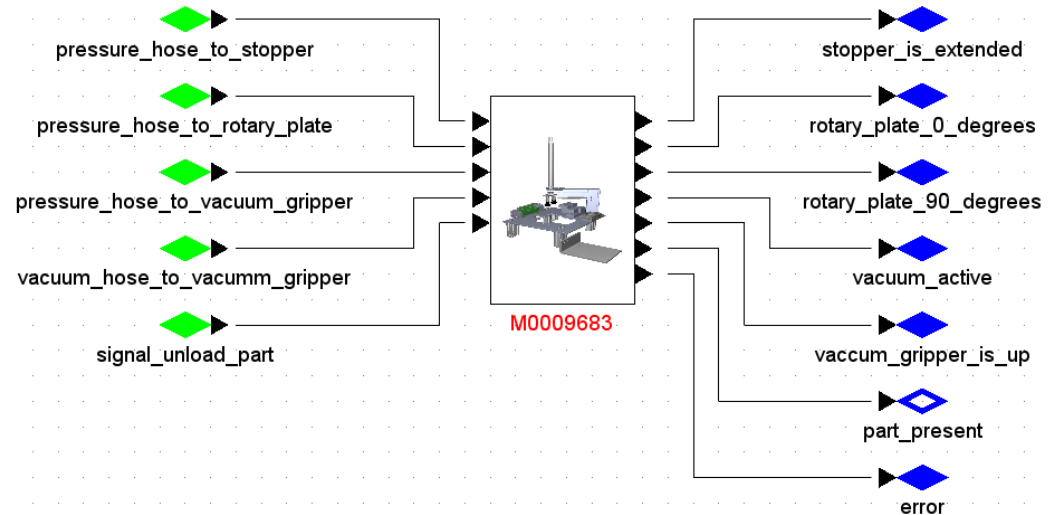


Figure 16. Simulation model *M0009683*, the pneumatic swivel unit, in the manner of a black box shown in ISG-virtuos[®].

Part of the inner structure of the black box model is shown in Figure 17. The screenshot has been labeled with the functionality behind the structure, *rotary plate movement*. Note, some of the green input and blue output ports can also be seen in Figure 16. The input *pressure_hose_to_rotary_plate* connected to the entire pneumatic swivel unit (Figure 16) delivers the medium (or signal) to the substructure (Figure 17) to rotate the plate by 90°. This will result in the same movement by the linked swivel arm. Feedback towards other simulation structures comes from the output ports (*rotary_plate_0_degrees*, *rotation*, *rotary_plate_90_degrees*). As shown in the substructure the rotation is realized with an integrator behavior allowing for a smooth movement.

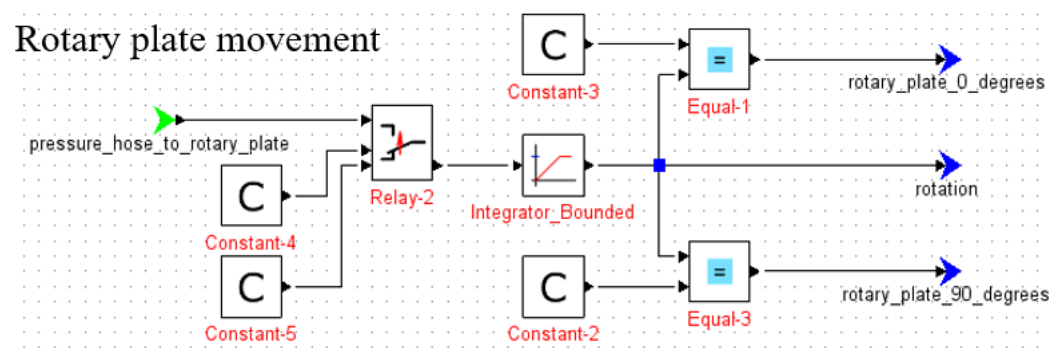


Figure 17. Detailed substructure of the black box simulation model showing the modeled movement of the rotary plate.

The purpose behind the detailed description of the simulation substructure is to show that all the instances, connections, constraints and parameters within the simulation model must be modeled in the EIPPM. Later, the simulation will be derived from the Design Graph and translated into an arbitrary simulation tool. This is the focus of the future work by the authors.

5. Discussion

In this section, the results, as well as the advantages and disadvantages of the EIPPM will be discussed in more detail. Since the presented approach partly emerged from a deficit analysis of the status quo in PS design (see Section 2.3), these deficits were deliberately addressed in the EIPPM. In addition, the embedding of a GBDL in the EIPPM and the intelligent linking with SBE is a key aspect of the presented approach.

A major advantage over other approaches is that a simulation model can be derived i.e., generated automatically from the EIPPM and executed at any time. This enables early and iterative use of simulations at different levels of detail (MiL, SiL or HiL) for validation of the current model stage. At the same time, validation by simulation can provide feedback that can lead directly to a modification of the boundary conditions and thus initiate a new design cycle with adjusted boundary conditions.

Another advantage of the EIPPM is that the data of the complete product-PS design is united in a central and interlinked data model, the design graph. In consequence, all disciplinary models are always consistent with each other, due to the automated generation from a common source. Domain-specific models (e.g., simulation models) can be generated in the appropriate format via interfaces from the design graph, see Figure 2.

Third, since the product is also modeled in the EIPPM, the dependencies between product and PS can be mapped directly. Changes to the product will therefore be taken into account immediately in the PS design. Within the EIPPM there are static components that do not change at runtime. These include the class diagram(s), the activity diagram(s), the individual rules and the interfaces. However, the activity diagram can contain elements that branch the flow depending on the precondition and the rules can also be formulated in such a way that different constraints lead to a different result. So overall, the design graph—i.e., the actual representation of a design cycle—is individually generated during execution. The design graph can then be translated at any time via interfaces into the target language of the desired domain.

The executability of the EIPPM, which it owes to its implementation as a GBDL makes a design of experiment possible that can be performed by changing the boundary conditions of the product and/or the production. This can not only lead to an optimization of the interaction between product and production, but also unconventional solutions may be considered that would otherwise never have been accounted for in the rigid models of the current status quo. As to the nature of this variant creation, there are two possible cases. In the first and simpler case of the two, they are merely parametric variants. These are characterized by the fact that the structure of the design graph remains constant and only the values of the attributes change. An example for such a parametric variant would be a change of the dimensions (e.g., additional 10 cm in length) of a component that remains constant in its shape (e.g., cubic). The second kind are topological variants, which are well supported by the re-executable sequence of design rules which directly manipulate the design graph as the abstract data model of the topology of the product and the production means.

As for the scalability of the approach, it should be noted that the Design Cockpit 43[®] can cope with very large models. Graphs with several hundred thousand nodes can be processed on a standard computer without any problems. The approach can also handle nested models. This plays a role especially for more complex products, which may consist of partial assemblies with their own PS. Here, the ability of GBDLs to modularize comes into play, allowing different projects to be combined into one. A recent publication on the topic of system of systems in GBDLs using the example of a production line and the associated building can be found in [84].

A disadvantage of the methodology is the increased modeling effort—creation of the class diagrams, rules, activities and interfaces—an activity which must be undertaken before the first designs can be generated. However, all the subsequently necessary models can then be generated automatically. In this creation process, the generalization effects (i.e., the existence of parametric and topological variants) must be taken into account.

For the creation of a single product-production model, this effort exceeds that of modeling with the tools and procedures of the status quo. In [85], this trade off between modeling effort and benefit from automation in knowledge-based engineering is discussed. The authors conclude that knowledge-based approaches—as the EIPPM also represents one—are, however, superior to manual ones in the long run. This finding at least gives hope that the EIPPM can ultimately reduce time-to-market. In principle, this would require a comparison of classical modeling with the EIPPM approach. However, this would go well beyond the scope of this paper, which is concerned with presenting the systematics. This issue will have to be addressed in future studies.

6. Summary and Conclusions

In this work, a new approach for the engineering of PSs was presented. The motivation for a new way of designing PS, also taking product requirements into consideration, stems from multiple shortcomings in current PS engineering. The authors name the exclusion of solution variants and discipline-specific decision making, as the two most disadvantageous actions. These disadvantages come from the predominant engineering process model that is mostly sequential and without feedback loops. These identified problems are not new to the research community and the most prominent way to address the shortcomings is seen in MBSE approaches (see Sections 2.1 and 2.3).

The authors introduced the fundamentals of VCOM and GBDL, as these topics are the base for the new engineering approach. With that knowledge and the given disadvantages, specific requirements were derived to conduct a literature review. From the literature review, the authors concluded that (a) none of the methods in the reviewed papers met all four requirements defined as indispensable for future automation in the context of this work, (b) the state of the art showed that model-based description of either products or PSs had been extensively researched, (c) the state of the art focused on facilitating the manual creation and handling of a central data model representing the PS or product.

The new approach, called *Executable Integrative Product-Production Model (EIPPM)*, uses a central data model but fundamentally changes the creation and update mechanisms of the model through automation. The automated creation and update, as well as transformation to discipline-specific models and views, is enabled through a formal description of vocabulary and rules. Vocabulary and rules are executable with a translation step of a compiler. Combined with a generation of simulation models for verification and validation tasks engineers are enabled to view and interact with the current state of the solution at any time. The feedback from engineering experts is taken back into vocabulary or rules to update the system design model. Requirements, from customer and the product to be produced on the PS are all modeled and thus taken into account.

To demonstrate the current state of the work progress an example PS from a laboratory course is used to show how the methodology can be applied in practice.

In the eyes of the authors, the presented work brings us a decisive step closer to the ultimate goal of fully automated design of PSs. Nevertheless, there is still a long way to go. The implementation of the automatic generation of virtual validation setups, based on a meta-simulation model is the consequent next step that has to be done. Also, the EIPPM will have to be extended to map further applications. Finally, the procedure behind EIPPM will have to be described in the form of a manual to be widely used.

Author Contributions: Conceptualization, D.S., K.K., S.R. and O.R.; methodology, D.S., K.K., S.R. and O.R.; software, D.S. and K.K.; writing—original draft preparation, D.S. and K.K.; writing—review and editing, D.S., K.K., S.R. and O.R.; visualization, D.S. and K.K.; supervision, S.R. and O.R. All authors have read and agreed to the published version of the manuscript.

Funding: Part of this work of the first author is supported by a grant from the European Regional Development Fund and the Ministry of Science, Research and the Arts of Baden-Württemberg, Germany (more information: <https://efre-bw.de/> (accessed on 25 May 2021)), in the context of the ZAFH-Project “Digitaler Produktlebenszyklus (DiP)” (grant No. 43031423).



EUROPÄISCHE UNION
Europäischer Fonds für regionale Entwicklung



Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

CAD	Computer-Aided Design
DSL	Domain Specific Language
EIPPM	Executable Integrative Product-Production Model
GBDL	Graph-based Design Language
HiL	Hardware-in-the-Loop
LHS	Left-Hand Side
IMS	Industrial Mechatronic System
MBE	Model-Based Engineering
MBSE	Model-Based Systems Engineering
MOF	Meta Object Facility
MiL	Model-in-the-Loop
PS	Production System
RHS	Right-Hand Side
SBE	Simulation-Based Engineering
SiL	Software-in-the-Loop
UML	Unified Modeling Language
VCOM	Virtual Commissioning

References

1. ElMaraghy, H.A. Flexible and reconfigurable manufacturing systems paradigms. *Int. J. Flex. Manuf. Syst.* **2005**, *17*, 261–276. [[CrossRef](#)]
2. Acerbi, F.; Sassanelli, C.; Terzi, S.; Taisch, M. A Systematic Literature Review on Data and Information Required for Circular Manufacturing Strategies Adoption. *Sustainability* **2021**, *13*, 2047. [[CrossRef](#)]
3. Royce, W.W. Managing the development of large software systems. In Proceedings of the IEEE WESCON 26, Los Angeles, CA, USA, 25–28 August 1970; TRW, Ed.; 1970; pp. 1–9.
4. Verein Deutscher Ingenieure e.V. *Digital Factory: Digital Factory Operations*; VDI Guideline 4499, Sheet 2; Verlag des Vereins Deutscher Ingenieure: Dusseldorf, Germany, 2011.
5. Kübler, K.; Scheifele, S.; Scheifele, C.; Riedel, O. Model-Based Systems Engineering for Machine Tools and Production Systems (Model-Based Production Engineering). *Procedia Manuf.* **2018**, *24*, 216–221. [[CrossRef](#)]
6. Verein Deutscher Ingenieure e.V. *Design of Technical Products and Systems: Model of Product Design*; VDI Guideline 2221, Sheet 1; Verlag des Vereins Deutscher Ingenieure: Dusseldorf, Germany, 2019.
7. Röck, S. Hardware in the loop simulation of production systems dynamics. *Prod. Eng.* **2011**, *5*, 329–337. [[CrossRef](#)]
8. Stöppler, G.; Menzel, T.; Douglas, S. Hardware-in-the-loop simulation of machine tools and manufacturing systems. *IEE Comput. Control Eng.* **2005**, *16*, 10–15. [[CrossRef](#)]
9. Reinhart, G.; Wunsch, G. Economic application of virtual commissioning to mechatronic production systems. *Prod. Eng.* **2007**, *1*, 371–379. [[CrossRef](#)]
10. Verein Deutscher Ingenieure e.V. *Digital Factory: Fundamentals*; VDI Guideline 4499, Sheet 1; Verlag des Vereins Deutscher Ingenieure: Dusseldorf, Germany, 2008.
11. Kühne, T. Matters of (Meta-) Modeling. *Softw. Syst. Model.* **2006**, *5*, 369–385. [[CrossRef](#)]

12. Atkinson, C.; Kuhne, T. Model-driven development: A metamodeling foundation. *IEEE Softw.* **2003**, *20*, 36–41. [[CrossRef](#)]
13. Caggiano, A.; Caiazzo, F.; Teti, R. Digital Factory Approach for Flexible and Efficient Manufacturing Systems in the Aerospace Industry. *Procedia CIRP* **2015**, *37*, 122–127. [[CrossRef](#)]
14. Neyrinck, A.; Lechler, A.; Verl, A. Automatic Variant Configuration and Generation of Simulation Models for Comparison of Plant and Machinery Variants. *Procedia CIRP* **2015**, *29*, 62–67. [[CrossRef](#)]
15. Yemencioğlu, E. Data Exchange for the Physics-Based Simulation of Material Handling Systems in the Digital Factory. Ph.D. Thesis, Otto-von-Guericke-Universität, Magdeburg, Germany, 2016.
16. Kübler, K.; Oberle, M.; Verl, A.; Riedel, O. Simulation-assisted run-to-run control for battery manufacturing in a cloud environment. In Proceedings of the 2017 24th International Conference on Mechatronics and Machine Vision in Practice (M2VIP), Auckland, New Zealand, 21–23 November 2017; pp. 1–6. [[CrossRef](#)]
17. Novák, P.; Ekaputra, F.J.; Biffl, S. Generation of Simulation Models in MATLAB-Simulink Based on AutomationML Plant Description. *IFAC-PapersOnLine* **2017**, *50*, 7613–7620. [[CrossRef](#)]
18. Martinez, G.S.; Sierla, S.; Karhela, T.; Vyatkin, V. Automatic Generation of a Simulation-Based Digital Twin of an Industrial Process Plant. In Proceedings of the IECON 2018—44th Annual Conference of the IEEE Industrial Electronics Society, Washington, DC, USA, 21–23 October 2018; pp. 3084–3089. [[CrossRef](#)]
19. Scheifele, C.; Verl, A.; Riedel, O. Real-time co-simulation for the virtual commissioning of production systems. *Procedia CIRP* **2019**, *79*, 397–402. [[CrossRef](#)]
20. Schopper, D.; Rudolph, S. From Model-Driven Architecture and Model-Based Systems Engineering via Formal Concept Analysis to Graph-Based Design Languages and Back: A Scientific Discourse. In Proceedings of the ASME 2018 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference IDETC/CIE 2018, Québec City, QC, Canada, 26–29 August 2018.
21. Bajaj, M.; Backhaus, J.; Walden, T.; Waikar, M.; Zwemer, D.; Schreiber, C.; Issa, G.; Martin, L. Graph-Based Digital Blueprint for Model Based Engineering of Complex Systems. *INCOSE Int. Symp.* **2017**, *27*, 151–169. [[CrossRef](#)]
22. Fricke, E.; Gebhard, B.; Negele, H.; Igenbergs, E. Coping with changes: Causes, findings, and strategies. *Syst. Eng.* **2000**, *3*, 169–179. [[CrossRef](#)]
23. Verein Deutscher Ingenieure e.V.; Verband der Elektrotechnik Elektronik Informationstechnik e.V. *Virtual Commissioning: Model Types and Glossary*; VDI/VDE Guideline 3693, Sheet 1; Verlag des Vereins Deutscher Ingenieure: Dusseldorf, Germany, 2016.
24. Rosen, R.; Jäkel, J.; Barth, M.; Stern, O.; Schmidt-Vollus, R.; Heinzerling, T.; Hoffmann, P.; Richter, C.; Puntel Schmidt, P.; Scheifele, C. *Simulation und Digitaler Zwilling im Engineering und Betrieb Automatisierter Anlagen—Standpunkte und Thesen des GMA FA 6.11*; VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik, Ed.; Automation 2019; VDI Verlag GmbH: Düsseldorf, Germany, 2019; pp. 531–560.
25. Oppelt, M.; Hoernicke, M.; Rosen, R.; Urbas, L.; Barth, M. *Simulation 2025: Simulation im Lebenszyklus Industrieller Anlagen*; VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik, Ed.; Automation 2016; VDI Verlag GmbH: Düsseldorf, Germany, 2016.
26. Scheifele, S.; Riedel, O.; Pritschow, G. Engineering of machine tools and plants using cyber-physical systems. In Proceedings of the 2017 Winter Simulation Conference, Las Vegas, Nevada, USA, 3–6 December 2017; IEEE Press: Piscataway, NJ, USA, 2017; pp. 1503–1514.
27. Xu, L. Wiederverwendbare Modelle zur Maschinensimulation für den Steuerungstest. Ph.D. Thesis, Technische Universität München, München, Germany, 2003.
28. Oppelt, M.; Wolf, G.; Drumm, O.; Lutz, B.; Baudisch, T.; Wehrstedt, J.C.; Krause, A.; Urbas, L. *Automatische Generierung von Simulationsmodellen für die virtuelle Inbetriebnahme auf Basis von Planungsdaten*; VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik, Ed.; Automation 2014; VDI-Verlag: Düsseldorf, Germany, 2014.
29. Barth, M. *Automatisch Generierte Simulationsmodelle Verfahrenstechnischer Anlagen für den Steuerungstest: Zugl.: Hamburg, Univ. der Bundeswehr, Diss*, 2011th ed.; Fortschritt-Berichte VDI Reihe 20, Rechnerunterstützte Verfahren; VDI-Verlag: Düsseldorf, Germany, 2011; Volume 438.
30. Barth, M.; Fay, A. Automated generation of simulation models for control code tests. *Control Eng. Pract.* **2013**, *21*, 218–230. [[CrossRef](#)]
31. Puntel Schmidt, P. Methoden zur Simulationsbasierten Absicherung von Steuerungscode Fertigungstechnischer Anlagen. Ph.D. Thesis, Helmut-Schmidt-Universität, Universität der Bundeswehr and Helmut-Schmidt-Universität, Hamburg, Germany, 2017.
32. Kufner, A. Automatisierte Erstellung von Maschinenmodellen für die Hardware-in-the-Loop-Simulation von Montagemaschinen. Ph.D. Thesis, Universität Stuttgart, Stuttgart, Germany, 2012. [[CrossRef](#)]
33. Weyrich, M.; Steden, F. Produktionssysteme modulbasiert simulieren: Methodische Identifikation wiederverwendbarer Simulationsmodule im Engineering-Prozess. *Wt Werkstattstech. Online* **2013**, *103*, 162–167.
34. Lindworsky, A. Teilautomatische Generierung von Simulationsmodellen für den entwicklungsbegleitenden Steuerungstest. Ph.D. Thesis, Technische Universität München, München, Germany, 2011.
35. Rudolph, S. *Übertragung von Ähnlichkeitsbegriffen*; Universität Stuttgart: Stuttgart, Germany, 2002.
36. Rudolph, S.; Kröplin, B. Entwurfsgrammatiken—Ein Paradigmenwechsel. *Prüfingenieur* **2005**, *26*, 34–43.
37. INCOSE Technical Operations. INCOSE Systems Engineering Vision 2020. Available online: http://www.cose.org/media/upload/SEVision2020_20071003_v2_03.pdf. (accessed on 25 May 2021)

38. Walter, B.; Martin, J.; Schmidt, J.; Dettki, H.; Rudolph, S. Executable State Machines Derived from Structured Textual Requirements—Connecting Requirements and Formal System Design. In Proceedings of the 7th International Conference on Model-Driven Engineering and Software Development, SCITEPRESS—Science and Technology Publications, Prague, Czech Republic, 20–22 February 2019; pp. 195–202. [CrossRef]
39. Rudolph, S.; Beichter, J.; Eheim, M.; Hess, S.; Motzer, M.; Weil, R. On Multi-Disciplinary Architectural Synthesis and Analysis of Complex Systems with Graph-based Design Languages. In Proceedings of the 62th Deutscher Luft- und Raumfahrtkongress (DGLR 2013), Stuttgart, Germany, 10–12 September 2013.
40. Groß, J.; Rudolph, S. Regelbasierte Analyse von Entscheidungen im Entwurf komplexer Systeme. In *Tag des Systems Engineering*; Maurer, M.; Schulze, S.O., Eds.; Carl Hanser Verlag GmbH & Co. KG: München, Germany, 2013; pp. 175–183. [CrossRef]
41. Groß, J.; Rudolph, S. Dependency Analysis in Complex System Design using the FireSat example. *INCOSE Int. Symp.* **2012**, *22*, 1856–1869. [CrossRef]
42. Kiesel, M.; Klimant, P.; Beisheim, N.; Rudolph, S.; Putz, M. Using Graph-based Design Languages to Enhance the Creation of Virtual Commissioning Models. *Procedia CIRP* **2017**, *60*, 279–283. [CrossRef]
43. Zech, A.; Stetter, R.; Till, M.; Rudolph, S. Automated generation of clamping concepts and assembly cells for car body parts for the digitalization of automobile production. In Proceedings of the Stuttgart Conference on Automotive Production (SCAP2020), Stuttgart, Germany, 9–10 November 2020.
44. Feldhusen, J.; Grote, K.H. (Eds.) *Pahl/Beitz Konstruktionslehre: Methoden und Anwendung Erfolgreicher Produktentwicklung*, 8 Vollständig Überarbeitete Auflage ed.; Springer: Berlin/Heidelberg, Germany, 2013.
45. BITKOM e.V.; VDMA e.V.; ZVEI e.V. (Eds.) *Umsetzungsstrategie Industrie 4.0: Ergebnisbericht der Plattform Industrie 4.0*. 2015. Available online: <https://www.bitkom.org/sites/default/files/file/import/150410-Umsetzungsstrategie-0.pdf> (accessed on 25 May 2021)
46. Groß, J.; Rudolph, S. Modeling graph-based satellite design languages. *Aerosp. Sci. Technol.* **2016**, *49*, 63–72. [CrossRef]
47. Groß, J.; Rudolph, S. Rule-based spacecraft design space exploration and sensitivity analysis. *Aerosp. Sci. Technol.* **2016**, *59*, 162–171. [CrossRef]
48. Groß, J.; Rudolph, S. Geometry and simulation modeling in design languages. *Aerosp. Sci. Technol.* **2016**, *54*, 183–191. [CrossRef]
49. Arnold, P.; Rudolph, S. Bridging the gap between product design and product manufacturing by means of graph-based design languages. In *Proceedings of the TMCE 2012, Karlsruhe, Germany, 7–11 May 2012*; Horváth, I., Ed.; Delft University of Technology: Delft, The Netherlands, 2012; pp. 985–998.
50. Kübler, K.; Schopper, D.; Riedel, O.; Rudolph, S. Towards an Automated Product-Production System Design—Combining Simulation-based Engineering and Graph-based Design Languages. *Procedia Manuf.* **2020**, *52*, 258–265. doi:10.1016/j.promfg.2020.11.043. [CrossRef]
51. Ingenieurgesellschaft für Intelligente Lösungen und Systeme mbH (IILS). The Design Cockpit 43. 2021. Available online: <https://www.iils.de/> (accessed on 25 May 2021).
52. Rabe, M.; Anacker, H.; Westermann, T.; Dumitrescu, R. Potential of using model-based systems engineering to improve the development process of engineering-to-order products in the field of machinery and plant engineering. *J. Teknol.* **2015**, *76*. [CrossRef]
53. Bursac, N.; Albers, A.; Ölschläger, M. Baukastenentwicklung durch MBSE am Beispiel einer modularen Fertigungsanlage im Kontext der Industrie 4.0. In *Tag des Systems Engineering, Herzogenaurach, 25–27 Oktober 2016*; Schulze, S.O., Tschirner, C., Kaffenberger, R., Ackva, S., Eds.; Hanser: München, Germany, 2017; pp. 247–256.
54. Wortmann, A.; Barais, O.; Combemale, B.; Wimmer, M. Modeling languages in Industry 4.0: An extended systematic mapping study. *Softw. Syst. Model.* **2019**, *86*, 997. [CrossRef]
55. Dziwok, S.; Pohlmann, U.; Piskachev, G.; Schubert, D.; Thiele, S.; Gerking, C. The MechatronicUML Design Method: Process and Language for Platform-Independent Modeling. Available online: <http://www.mechatronicuml.org/> (accessed on 25 May 2021).
56. Pohlmann, U. A Model-Driven Software Construction Approach for Cyber-Physical Systems. Ph.D. Thesis, Universität Paderborn, Paderborn, Germany, 2018, doi:10.17619/UNIPB/1-313. [CrossRef]
57. Alvarez Cabrera, A.A.; Foeken, M.J.; Tekin, O.A.; Woestenenk, K.; Erden, M.S.; de Schutter, B.; van Tooren, M.; Babuška, R.; van Houten, F.; Tomiyama, T. Towards automation of control software: A review of challenges in mechatronic design. *Mechatronics* **2010**, *20*, 876–886. [CrossRef]
58. Hackenberg, G.; Richter, C.; Zäh, M. *IMoMeSA—Abschlussbericht: Integrierte Modellbasierte Entwicklung Mechatronischer Systeme im Maschinen- und Anlagenbau*; Abschlussbericht; Technische Universität München: München, Germany, 2015.
59. Hackenberg, G. Test-Driven Conceptual Design of Cyber-Physical Manufacturing Systems. Ph.D. Thesis, Technische Universität München, München, Germany, 2018.
60. Botaschanjan, J.; Hensel, T.; Hummel, B.; Lindworsky, A.; Zäh, M.F.; Reinhart, G.; Broy, M. *AutoVIBN—Abschlussbericht: Automatische Generierung von Verhaltensmodellen aus CAD-Daten für die Qualitätsorientierte virtuelle Inbetriebnahme*; Abschlussbericht; Technische Universität München: München, Germany, 2010.
61. Albers, A. Five hypotheses about engineering processes and their consequences. In Proceedings of the TMCE, Ancona, Italy, 12–16 April 2010.
62. Albers, A.; Muschik, S. The Role and Application of Activities in the Integrated Product Engineering Model (iPeM). In Proceedings of the International Design Conference—DESIGN 2010, Dubrovnik, Croatia, 17–20 May 2010; pp. 127–136.

63. Albers, A.; Reiss, N.; Bursac, N.; Richter, T. iPeM—Integrated Product Engineering Model in Context of Product Generation Engineering. *Procedia CIRP* **2016**, *50*, 100–105. [CrossRef]
64. Albers, A.; Fahl, J.; Hirschter, T.; Haag, S.; Hunemeyer, S.; Staiger, T. Defining, Formulating and Modeling Product Functions in the Early Phase in the Model of PGE—Product Generation Engineering. In Proceedings of the 2020 IEEE International Symposium on Systems Engineering (ISSE), Vienna, Austria, 12 October–12 November 2020; pp. 1–10. [CrossRef]
65. Mandel, C.; Wolter, K.; Bause, K.; Behrendt, M.; Hanf, M.; Albers, A. Model-Based Systems Engineering methods to support the reuse of knowledge within the development of validation environments. In Proceedings of the 2020 IEEE International Systems Conference (SysCon), Montreal, QC, Canada, 24 August–20 September 2020; pp. 1–7. [CrossRef]
66. Bursac, N. Model Based Systems Engineering zur Unterstützung der Baukastenentwicklung im Kontext der Frühen Phase der Produktgenerationsentwicklung. Ph.D. Thesis, Karlsruher Institut für Technologie, Karlsruhe, Germany, 2016.
67. Object Management Group. OMG Meta Object Facility (MOF) Core Specification. Available online: <https://www.omg.org/spec/MOF/2.5.1/PDF> (accessed on 25 May 2021)
68. *Kombination und Integration Etablierter Technologien mit Additiven Fertigungsverfahren: Projekt-Abschlussbericht KitkAdd*; Karlsruher Institut für Technologie (KIT) wbk Institut für Produktionstechnik: 2020. Available online: <http://www.kitkadd.de/> (accessed on 25 May 2021). [CrossRef]
69. Jacob, A.; Windhuber, K.; Ranke, D.; Lanza, G. Planning, Evaluation and Optimization of Product Design and Manufacturing Technology Chains for New Product and Production Technologies on the Example of Additive Manufacturing. *Procedia CIRP* **2018**, 108–113. [CrossRef]
70. Jacob, A.; Steimer, S.; Stricker, N.; Häfner, B.; Lanza, G. Integrating product function design, production technology optimization and process equipment planning on the example of hybrid additive manufacturing. *Procedia CIRP* **2019**, *86*, 222–227. [CrossRef]
71. Eigner, M.; Koch, W.; Muggeo, C. (Eds.) *Modellbasierter Entwicklungsprozess Cybertronischer Systeme*; Springer: Berlin/Heidelberg, Germany, 2017; doi:10.1007/978-3-662-55124-0.
72. Oestersötebier, F. Modellbasierter Entwurf Intelligenter Mechatronischer Systeme Mithilfe Semantischer Technologien. Ph.D. Thesis, Universität Paderborn, Paderborn, Germany, 2018. [CrossRef]
73. Stoffels, P.; Kaspar, J.; Bähre, D.; Vielhaber, M. Integrated Product and Production Engineering Approach—A Tool-Based Method for a Holistic Sustainable Design, Process and Material Selection. *Procedia Manuf.* **2018**, *21*, 790–797. doi:10.1016/j.promfg.2018.02.185.
74. Verl, A.; Haubelt, A. Der Weg zur automatischen Generierung von Simulationsmodellen aus mechatronischen Baukästen. In *Effiziente Methodiken und Durchgängige Werkzeuge zur Modellerstellung Digitaler Produktionseinrichtungen*; Brecher, C., Ed.; Fortschritt-Berichte VDI Reihe 2, Fertigungstechnik 671; VDI Verlag: Düsseldorf, Germany, 2009; pp. 55–70.
75. Verl, A.; Müller, V.; Haubelt, A. Baukastenbasiertes simulationsgestütztes Engineering. *A&D-Kompendium* **2009**, *2009/2010*, 72–74.
76. Reuter, A.; Müller, V.; Verl, A. Disziplinübergreifendes Engineering: Integration von Simulationsdaten in mechatronische Komponentenmodelle. *Wt Werkstattstech. Online* **2010**, *100*, 399–406. [CrossRef]
77. Voß, V. Wiederverwendbare Simulationsmodelle für die domänen- und disziplinübergreifende Produktentwicklung. Ph.D. Thesis, Universität Stuttgart, Stuttgart, Germany, 2012. [CrossRef]
78. Brovkina, D.; Kienzlen, A.; Riedel, O. Comparative Analysis of Factory Simulation Description Models for Comprehensive Description of Model Design. In Proceedings of the 2020 IEEE Eurasia Conference on IOT, Communication and Engineering (ECICE), Yunlin, Taiwan, 23–25 October 2020, pp. 318–321. doi:10.1109/ECICE50847.2020.9301947.
79. Verein Deutscher Ingenieure e.V.; Verband der Elektrotechnik Elektronik Informationstechnik e.V. *Formalised Process Descriptions: Concept and Graphic Representation*; VDI Guideline 3682 Sheet 1; Verlag des Vereins Deutscher Ingenieure: Dusseldorf, Germany, 2015.
80. Verein Deutscher Ingenieure e.V. *Design Methodology for Mechatronic Systems*; VDI Guideline 2206; Verlag des Vereins Deutscher Ingenieure: Dusseldorf, Germany, 2004.
81. Meta-Modeling and the OMG Meta Object Facility (MOF): A White Paper by the OCUP 2 Examination Team. Available online: <https://www.omg.org/ocup-2/documents/Meta-ModelingAndtheMOF.pdf> (accessed on 25 May 2021)
82. Grönninger, H.; Krahn, H.; Pinkernell, C.; Rumpe, B. Modeling Variants of Automotive Systems using Views. In *Tagungsband Modellierungs-Workshop MBEFF: Modellbasierte Entwicklung von Eingebetteten Fahrzeugfunktionen*; TU Braunschweig: Braunschweig, Germany, 2008. [CrossRef]
83. Schmidt, J.P.; Zeller, A.; Weyrich, M. Modellgetriebene Entwicklung serviceorientierter Anlagensteuerungen. *At-Automatisierungstechnik* **2017**, *65*, 26–36. doi:10.1515/auto-2016-0107.
84. Voss, C.; Petzold, F.; Rudolph, S. Linking Building Design with the Digital Factory by Graph-based Design Languages. In Proceedings of the TMCE 2020, Dublin, Ireland, 11–15 May 2020. [CrossRef]
85. Esanakula, J.; Sridhar, N.V.; Rangadu, V. Knowledge Based Engineering: Notion, Approaches and Future Trends. *Am. J. Intell. Syst.* **2015**, *2015*, 1–17. doi:10.5923/j.ajis.20150501.01.

Short Biography of Authors



Dominik Schopper (born 1989) is a PhD student and member of the research group “Design Theory and Similarity Mechanics” at the Institute of Aircraft Design (IFB) at the University of Stuttgart. His scientific work focuses on Graph-based Design Languages, round-trip engineering and design automation. Mr. Schopper received his diploma (M.Sc.) from the University of Stuttgart in the course of study Aeronautics and Space Engineering.



Karl Kübler (born 1988) is head of the research group for “Virtual Methods in Production Engineering” at the Institute for Control Engineering of Machine Tools and Manufacturing Units (ISW) of the University of Stuttgart. His scientific work focuses on virtual commissioning, simulation-based engineering and test automation. Mr. Kübler received his diploma (Dipl.-Ing.) from the University of Stuttgart in the course of study Automation Technology in Production.



Stephan Rudolph (born 1961) holds various academic engineering degrees (Dipl.-Ing.), (Dr.-Ing.) and (Priv.-Doz.) from the University of Stuttgart, Germany. Stephan Rudolph is Head of the research group “Design Theory and Similarity Mechanics” at the Institute of Aircraft Design (IFB) at the University of Stuttgart and teaches several courses on digital engineering. His research interests include formal methods in Model-Based System Engineering (MBSE) and formal engineering design synthesis methods, automatic model generation and design evaluation methods. The second research interest are applications of similarity mechanics in engineering and artificial intelligence.



Oliver Riedel (born 1965) is Full Professor and Head of the Institute for Control Engineering of Machine Tools and Manufacturing Units (ISW) at the University of Stuttgart and holds the chair of production information technologies. He is also the institute director at Fraunhofer Institute for Industrial Engineering IAO in Stuttgart, where he focuses on digital engineering. Previously, he held several senior management positions in the IT and automotive industries. For more than 20 years, he has been working on the fundamentals and practical application of methods for virtual assurance in product development and production.