Proceedings of TMCE 2012, May 7–11, 2012, Karlsruhe, Germany, edited by I. Horváth and Z. Rusák

BRIDGING THE GAP BETWEEN PRODUCT DESIGN AND PRODUCT MANUFACTURING BY MEANS OF GRAPH-BASED DESIGN LANGUAGES

Peter Arnold
Institut für Statik und Dynamik der Luft- und Raumfahrtkonstruktionen
Universität Stuttgart
arnold@isd.uni-stuttgart.de

Stephan Rudolph
Institut für Statik und Dynamik der Luft- und Raumfahrtkonstruktionen
Universität Stuttgart
rudolph@isd.uni-stuttgart.de

ABSTRACT

By now there still exists a major gap in the seamless information integration between product design and manufacturing processes. The reasons for this gap are manifold, however the incompatibility between the modeling philosophy used in the product design and for product manufacturing as well as the inconsistency in the data models and formats used in both areas are among the most obvious. Bridging this gap can be achieved by the use of graph-based design languages. The approach is illustrated using an aircraft panel structure model which is incrementally mapped by a sequence of model-transformations into a model from which the digital factory which produces the designed panel structure can be generated automatically. This proves that topological and parametrical design changes in the design phases can be propagated automatically into a digital factory environment, where the effects of the design changes can be evaluated using appropriate time or cost metrics. According to the authors best knowledge this is the first time that the gap between product design and product manufacturing is digitally closed and that design changes and resulting manufacturing process updates are successfully automated for parametrical and topological changes in design and production.

KEYWORDS

Design language, model transformation, design automation, digital factory, manufacturing sequence

1. INTRODUCTION

The manufacturing of the product on the shop floor is economically the most important phase depending on the product development process since all design decision have a great impact on the subsequent value creation process. The digital factory is a tool to support the development of the real factory and a lot of digital models and simulations are available for this purpose. The product development phase is nowadays also largely supported by digital models. Currently the two fields (e.g. of design and production) are not really connected with each other and after significant changes in models in an early design stage a lot of models must be updated. Furthermore, if product design changes are taken seriously and involve not only parametrical but topological changes, conventional software tool suites with internal parametric-associative process chains reach their theoretical and practical limitations. As a consequence, often only few alternative design variants are compared due to tight time contraints. However, it is pointed out that it would be very useful if the impact of early design decisions (or changes) could be quantified in the factory.

To overcome the gap between product design and product manufacturing, the novel approach of graph-based design languages is used. The concept of design languages leans from natural languages spoken by humans in which a vocabulary and rules make up a grammar. This means that a valid sentence is a legal combination of vocabulary representing a valid design. A key aspect in such a design language is its graphical format in form of a graph-based design language based on the Unified Modeling Language (UML). The language aspect beeing 'graph-based' allows to modify topology (i.e. the question whether a certain object, property or behavior exists) can be changed as easy as the design parameters contained in a node of the graph. Design languages offer

therefore a combined manipulation scheme for design topology and parameters. In principle the approach consists of a central data model and a dedicated sequence of model transformations to incrementally change this model which ensures that the actual model is always 'up to date' and all changes along the design process chain can be automatically propagated downward. This 'front-loading' allows to investigate the effects of design and manufacturing alternatives in the digital factory.

The approach is illustrated using the example of the conceptual design phase for an aircraft structure element (e.g. panels) which is subsequently mapped by a sequence of model transformations into a model which allows to generate the digital factory which produces the designed structure automatically. Finally it is shown that significant topological and parametrical design changes in the conceptual design phase can be propagated automatically into a digital factory enviroment, where the effects of both the topological and parametrical design changes can be evaluated using time or cost metrics.

1.1. Graph-based desing languages

The concept of graph-based design languages (Rudolph, 2003) has evolved over the last ten years into a generic framework for the definition of computerized design processes. The corresponding design compiler (called design compiler 43) was developed by the IILS mbH (http://www.iils.de, 2012) in cooperation with the University of Stuttgart. The design compiler 43 generates design representations in analogy to the principle of compilation known from modern programming languages (see figure 1). In principle, design languages consist of a meta-model that stores all relevant parametrical and topological design information and acts as a centralized model repository. It uses a set of design rules and a design compiler that executes the design rules and automatically derives the analysis models of the respective design domains (CAD, CFD, FEM, ...). The formalisms of design languages are discussed in depth in (Alber & Rudolph, 2003) and (Rudolph, 2002). The applications in the automotive sector is shown in (Haq & Rudolph, 2004) and (Kormeier & Rudolph, 2006) and in the aerospace sector are presented in (Schäfer & Rudolph, 2004), (Irani & Rudolph, 2003) and (Gross & Rudolph, 2011).

As a meta-model to store the data for the so called graph-based design languages an international standardized modeling language, the Unified Modeling

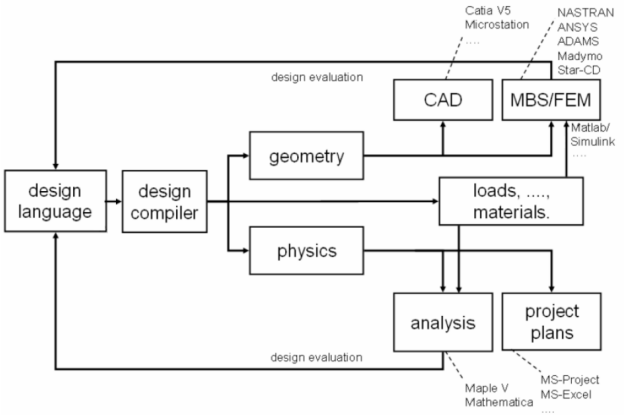

Figure 1 design compiler architecture (Rudolph, 2002)

Language (UML) (OMG, 2009), is used. The UML has been developed by software engineers and provides many features to model data to describe object oriented software. It also has extension mechanisms (so-called lightweight and heavyweight extensions) to extend the UML with domain specific aspects (Reichwein, 2011). System engineers have also defined a modeling standard, the Systems Modeling Language (SysML) (OMG, 2010), which has common modeling subsets with the UML and has some additional diagrams and modeling elements. However, the discussion of the pros and cons of SysML as a language alternative versus UML is not in the scope of the present paper.

To be able to reproduce the design process and the associated model changes, model transformations (OMG, 2003) are used and their sequence is modeled in a UML activity diagram.

1.2. Digital factory

The term 'digital factory' is defined as: *Digital factory is the generic term for a comprehensive network of digital models, methods and tools - including simulation and 3D visualisation - integrated by a continuous data management system.... Its aim is the holistic planning, evaluation and ongoing improvement of all structures, processes and resources of the real factory in conjunction with the product.* (VDI 4499, 2008)

Other definitions are given by (Bley & Franke, 2001), (Dombrowski et al., 2001), (Schuh et al., 2002), (Westkaemper et al., 2003), (Wiendahl, 2002). Essentially the individual definitions of the digital factory understand these as digital models that represent the relevant information of the real factory or as a

tool to create this model. They also define connections to the product development and various simulations which are also digital models.

In order to represent data in a model, a meta-model and a modeling language (meta-metamodel (Booch et al., 1999)) is needed. The advantages of an object-oriented modeling and the use of the UML in factory planning are shown in (Bergholz, 2005). A first version of a meta-model of the digital factory (figure 2) was defined by (Jonas, 1999). The conceptual possibilities of an automated planning of assembly processes are shown in (Cuiper, 2000). A good overview of possible modeling approaches and their pros and cons is presented in (Kapp, 2011). (Kuehn, 2006) suggested an 'Open Factory Backbone' to support the continuous integration of the factory simulation in the digital factory, but the concept and the underlying information model is not further specified. (VDI 3633-5, 2000) specifies the data link for a efficent simulation use and points out the need for such an IT-system that solves the conistency problem.

1.3. Problem statement

According to the definition stated in section 1.2 the digital factory consist of numerous digital models. There are basically two approaches to exchange data between different models (figure 3). The first possibility is that every model exchanges data with every other model but each additional model leads to a quadratic ($n*(n-1)$) rise of the needed interfaces. The second possibility is to use a central data model which contains all information. To exchange data between the central model and the domain models only $2*n$ interfaces are required. Moreover, because every model is derived through the central model, the consistency problem between the different models is solved elegantly. Additionally, the second approach is preferrable through the mathematical observation that there is no bijective mapping between vector spaces of different dimensionality. It exist only a transformation between higher spaces to lower spaces and so the central data model must be a superset of the domains which should be examined. See (Reichwein, 2011) for more information on the central data model topic.

Furthermore the general model theory by Stachowiak (Stachowiak, 1973) states that a model is a limited perception of the reality, in this case of the real factory. The characteristics of a model are described by the protection feature, the shortening feature and the pragmatic feature. The last one says that every model has a specific purpose at a specific time and answers a (single) question. Because the design process can be seen as a sequence of questions that should be answered, there is the need of different models at different times during the design cycle and moreover, it is an iterative process. Often recurrent routine activities must be executed and it is difficult to implement algorithms because the data is spread over different, often proprietary simulation models.

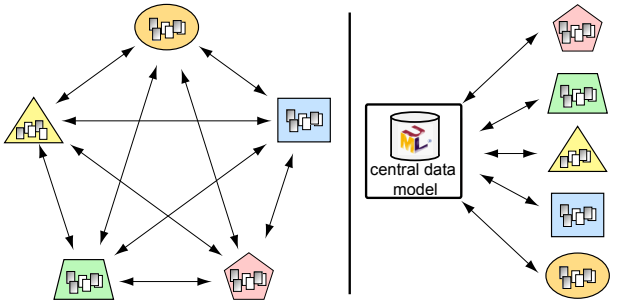

Figure 3 model exchange

1.4. Solution approach

In this paper, the method of graph-based design languages will be applied to the digital factory to try to address the existing problems. In particular to overcome the restriction of the numerous proprietary models the relevant data is kept in a central data model. To get the full benefit, it is also useful to use the same approach on the product development. This will be demonstrated by prior work on an airplane panel (Fuhr, 2010). To demonstrate the possibility of model transformations to describe engineering activities, a factory cell and a painting simulation will be used. Every time a change occurs it is possible to re-execute the transformations and to solve the consistency problem. Moreover, an algorithm to determine assembly sequences automatically will be presented.

2. PRIOR WORK

Many conceptual design studies have been done using graph-based design languages which also referenced in section 1.1. The product which is considered now in detail is described below.

2.1. Aircraft

The aircraft design was also target of various studies with design languages. One language has been build to generate the exterior skin of the airplane for use in a computational fluid dynamics (CFD) simu-

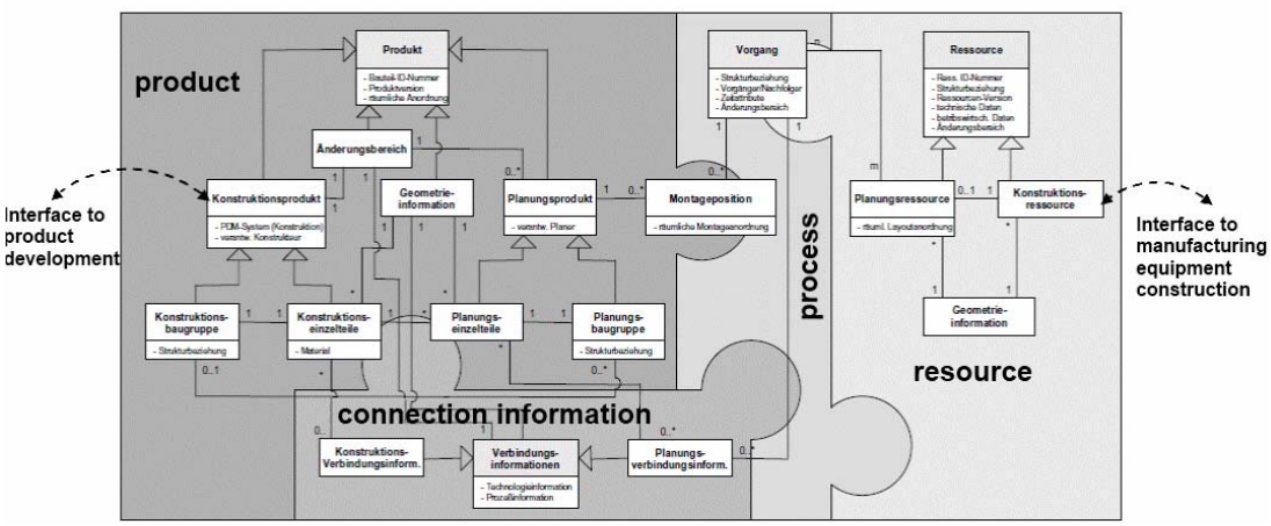

Figure 2 Meta-model digital factory (Jonas, 1999)

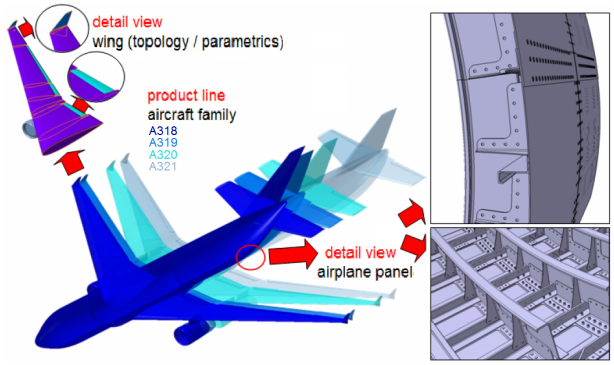

Figure 4 aircraft family design and panel details

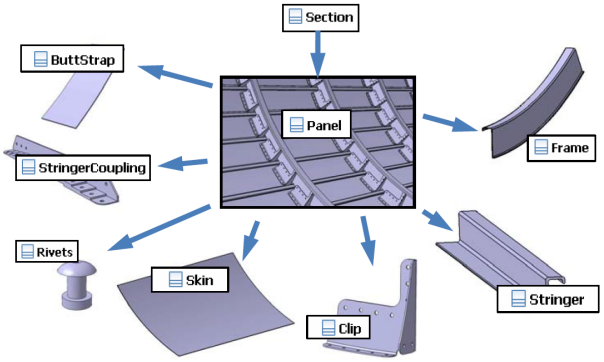

Figure 5 decomposition airplane panel (Fuhr, 2010)

lation (Böhnke, 2009). Another language has been created to generate structural components of a fuselage section (Fuhr, 2010) which is now outlined in more detail, see figure 4.

The shortening feature mentioned in section 1.3 states, that from the set of all attributes or elements of the original just those that are relevant for the specific modeling purpose need to be selected. A possible decomposition of an airplane panel is shown in figure 5. This is a geometric decomposition, but functional or any other kinds of decomposition principles are also possible. The meta-model for an airplane panel design can be created with these identified components.

After this, a design sequence must be defined. For example, to define the initial parameters, a dimensions concept needs to be calculated and the required components must be added. This sequence can be modeled as an activity diagram in the design language. The steps in this sequence modify the model and can be represented as model transformations. This way every design step is captured in digital form. Different kinds of transformations are possible, ranging from the simple addition of a component to the execution of complex algorithms. Once the process chain is established, various variants with different parameters and topology in CATIA (http://www.catia.com, 2012) (see figure 6) can be generated.

3. DIGITAL FACTORY META-MODEL

A meta-model for the digital factory from the literature was introduced in section 1.2. This meta-model was implemented in the UML and analyzed in detail. It turned out that it is well suited to model the major aspects of the central model for the digital factory. The class *Process* was extended with subclasses like *Handling*, *Joining*, etc. to describe the assembly process more precise (Grote & Feld-

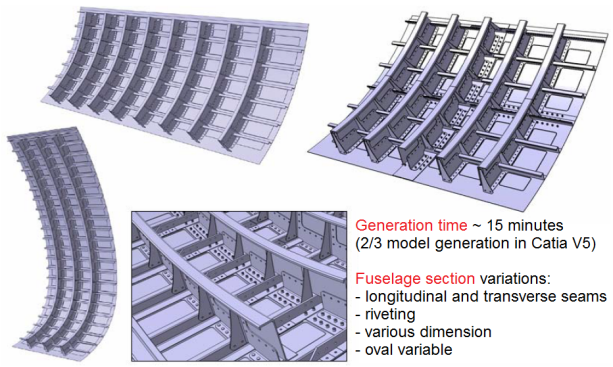

Figure 6 airplane panel variants (Fuhr, 2010)

husen, 2007) and because the path planning should also be included, the package Path (figure 7) was added. For this purpose the path planning meta-model of DELMIA (http://www.delmia.com, 2012) was implemented.

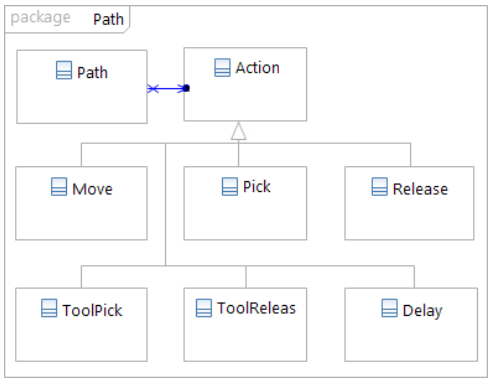

Figure 7 package 'Path' in the meta-model

The interface to the product development and manufacturing equipment creation works well. For example, on the product development side the class *Product* has an association to the *ConstructionProduct* class. The top class of the meta-model for the product inherits from this class. So the connection can be established and the accessibility of all relevant information of the product is given during the factory planning process.

The use of the meta-model is shown in figure 8. The product is represented during the planning process through an instance of the class *PlanningProduct*. This instance is linked to the corresponding instance in the product development that inherits from *ConstructionProduct*. If for example during the planning process information of the mass of the product is needed, it can be accessed through the interface.

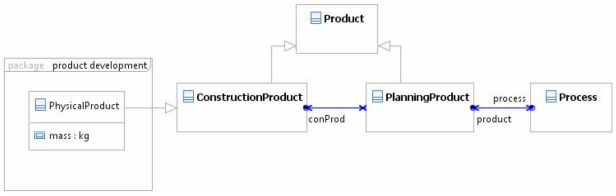

Figure 8 example interface to product development

4. FACTORY PLANNING

As seen in chapter 2.1 one can build up a model of a specific airplane panel relatively fast. Starting with the relevant panel information, a digital factory cell for the assembling can be designed and generated.

4.1. Factory sequence planning

For this purpose one variant of a planning sequence of the factory is modeled. Each step in figure 9 is again a sub-sequence leading to the following hierarchical sequence of steps.

The **first step** *importProductData* is to import the relevant product data and create the corresponding planning instances in the central model. Each part is linked to its construction part. Before a product model can be imported, some conditions must be satisfied. For example, the local product coordinate origin should be located in a useful position on the product because this will make the path planning easier. Since this is no issue in the airplane panel design phase a transformation is added to solve this issue.

The **second step** *addResourceAndLayout* adds the necessary resources and defines their relationship with respect to the product. The coordinates for the layout of the manufacturing cell are defined and can also be easily edited by an interface to a layout program. In this case, a simple java program, in which the coordinates are defined by simple drag and drop operations, is used. The same interface mechanism as described in section 3 for the product development will also be used for the resources. So first the planning instances are created and linked to abstract place holders for the resources. The resources will be selected on runtime and can also be modeled as a graph-based design language with the products as 'input'. This way the resources can be adapted to the actual product configuration.

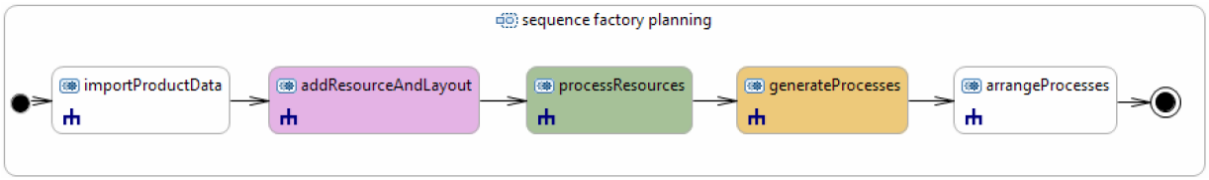

Figure 9 Factory sequence planning

In the **third step** *processResources* the resources will be processed. First, the main assembly group will be placed on the corresponding resource and the position of all sub-assembly groups and products will be calculated. Second, all products will be placed on their resources. The mounting position is stored and an instance for the specific assembly process for each product is created. Finally, the robots will be selected. For this purpose a certain kind of library (figure 10) was created. Because at this point in the planning all products are placed on the resources and all mounting position are known, the maximum ranges and the maximum loads for the robots are defined. With this input, a specific robot can be selected (figure 10 and 11). In this case KUKA robot models are used in DELMIA. Once a robots is selected, a rule (figure 12) replaces the abstract placeholder instance with a specific one.

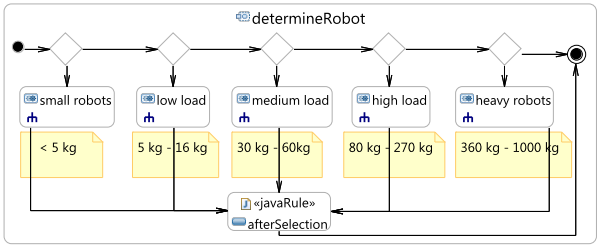

Figure 10 robot library

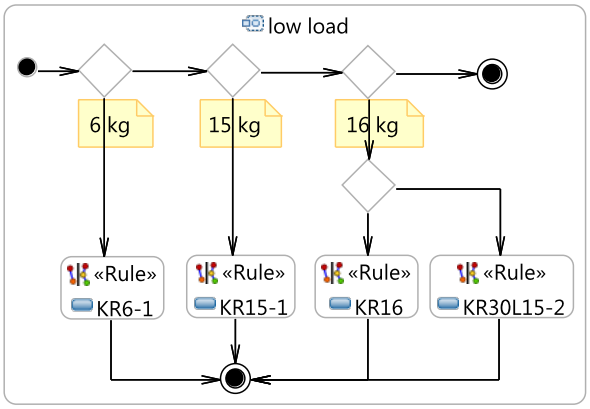

Figure 11 robot library subprogram

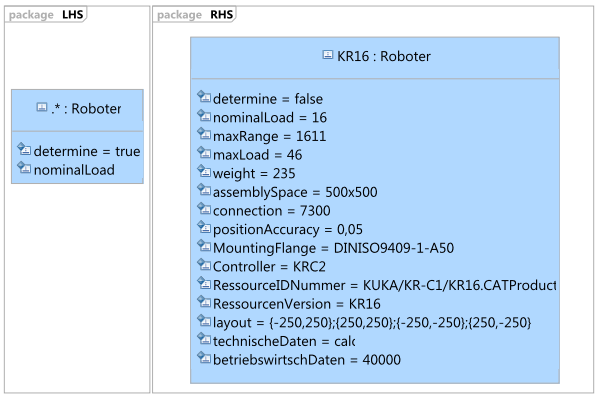

Figure 12 robot KUKA KR16 rule

In the **fourth step** *generateProcesses* all processes will be generated and all operations and paths will be calculated automatically. For more information on the generation of the Path Planning see chapter 4.2. First, the 'pick and tool release' process with its paths is generated. Next, the handling and the joining processes will be created as described in section 4.2.

The **last step** *arrangeProcesses* consist of the assembly sequence calculation described in chapter 4.3.

4.2. Path planning

The path planning is an important part of the factory planning. As described in chapter 3, the process is divided in several sub-steps. Only the handling and the joining steps are considered. Before a part can be joined, it must be first placed in the correct position for the handling process. This process can be divided in three parts. The first sub-path from the resource to a position in the close-up range is provided by the resource itself. According to the object-oriented paradigm '*loose coupling and a strong bond (cohesion)*' all information for this sub-path should be clustered with the resources. This way a resource can easily be exchanged. The second sub-path goes from this position in the close-up range of the re-

source to a position near the final product position. In this stage the shortest path is chosen and (for the moment) no collision detection is undertaken since this feature is provided later by the domain model in DELMIA. The last sub-path is the final product positioning. Because this is product specific and not the main aim of the paper, no generic claim for this approach is made. Nevertheless, the following method has proven to be practical and robust in this case study. Starting at the final product position which is known, all path position can be calculated relative to it. This way it is easy to reuse the paths if there are many similar parts. The joining paths are generated in the same way. Each of these steps needs a tool to handle the part accordingly. Thus the paths to pick and release the tool are also needed. For this purpose the same approach as for the resources is used. In this way the tools can also be changed without further effort.

4.3. Assembly sequence

The assembly sequence determination is an important component of the factory planning process. Therefore a lot of research has been done already on this topic. The algorithms can roughly be divided into two classes of the 'recursive decomposition of the assembly' and the 'mathematically oriented method of graph theory'. Before the assembly sequence can be extracted through decomposition, the product parts must be combined into the product in the composition process, so this approach often seems a bit long-winded. An overview about graph-based algorithms contains (Whitney et al., 1999).

Basically both approaches need a set of constraints and compute all possible assembly sequences for this set. But this is not feasible for a large number of assembly parts because either many conditions have to be set or a very large number of sequences are generated. Especially in cases where parts can be assembled simultaneously the number of sequences generated explodes with n!. For example, in the airplane panel assembly, the clips in a row can be assembled in parallel and for 6 clips this would give 6!=720 sequences per row which is not very manageable. Therefore we use a graph algorithm that is normally used for generation of the solution path for a mathematically constraint system. Mathematical equations express correlations between variables and part connections can be expressed accordingly. More information on the algorithm can be found in (Serrano, 1987). In figure 13 you can see a graphical user interface (GUI) for the assembly sequence algorithm. On the left side you see a list of all assembly operations and the parts are listed on the top side. In the editor area, the conditions can be defined which assembly operation couple with which part. For this group of conditions the assembly sequence is calculated immediatedly on the right side. The constraints for the airplane panel are shown, which can be determined directly from the product model, for example: 'stringer is connected to clip 1-5' (figure 14). This set of constraints not yet determines a unique sequence so further constraints are needed. Such additional constraints cannot be determined easily from the product development because these constraints are mainly manufacturing conditions. The maximal parallel assembly sequence can be determined after these additional constraints are added. For the airplane panel example the solution given in Figure 13 can be calculated. Here one can see that all clips and frames can be assembled interchangeably. In this way they can be assembled sequentially or simultaneously or all combinations in between. This is dependent on how many assembly robots or machines are available and useful. An algorithm that only gives all possible assembly sequences would have to calculate 30! solutions, which are about ~10^{32} options, so this is not feasible in practice.

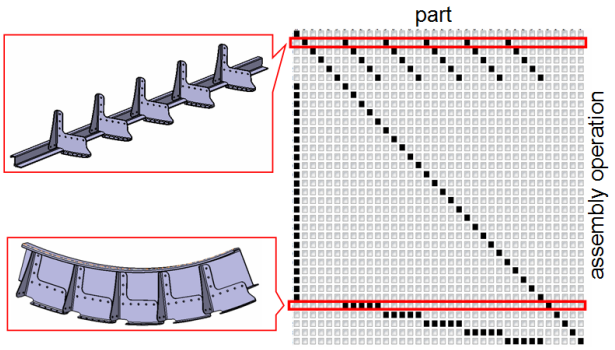

Figure 14 only mating conditions

By exchanging groups of conditions, many sequence variations can be created. If possible, the algorithm always returns a solution. For example in figure 15, the sequence of the assembly of the clips or the frames can be inverted by mirroring of the constraints above or below the diagonal.

4.4. Domain models

The generated central data model with all necessary input and calculations can now be transformed in domain specific models. A transformation to DELMIA

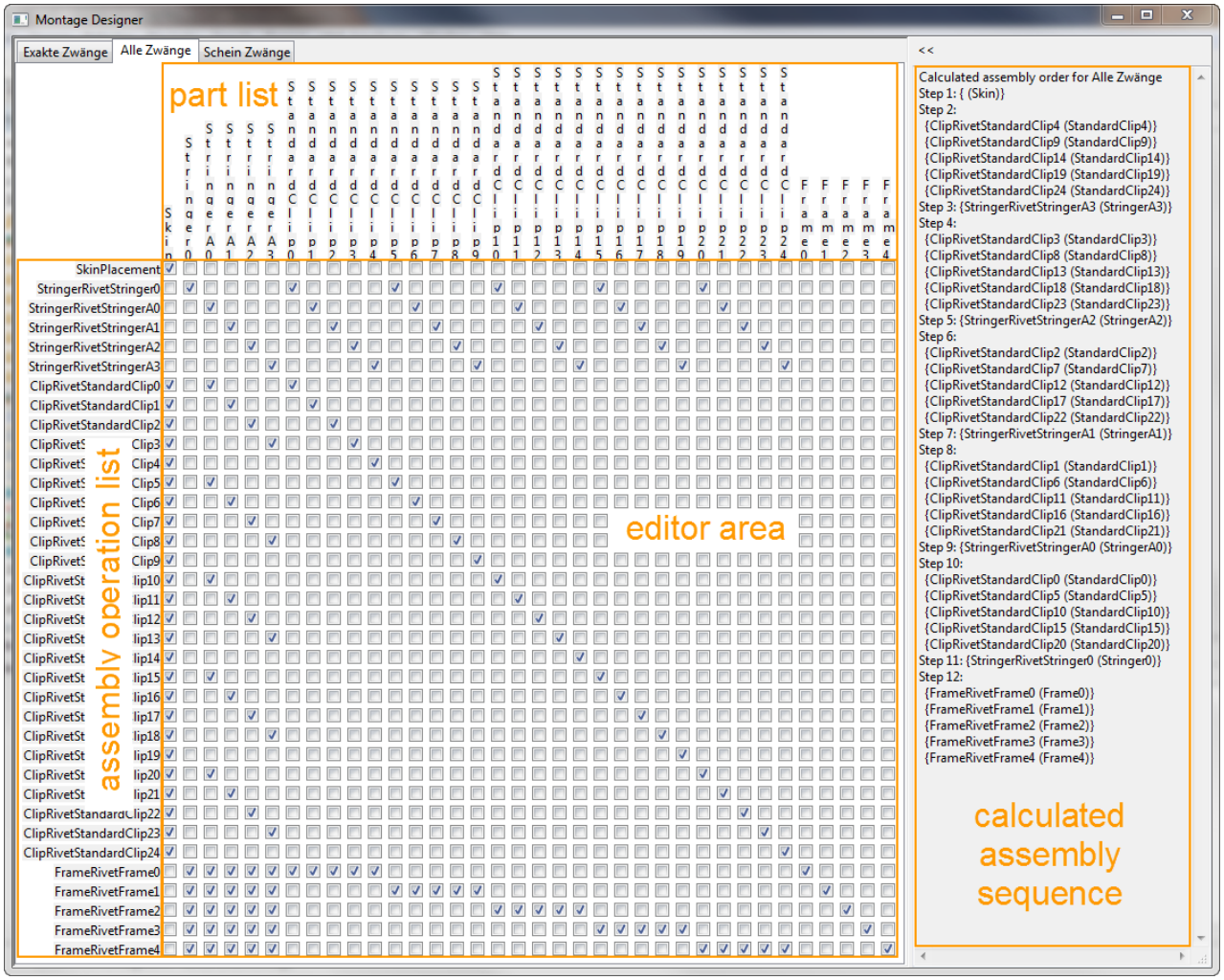

Figure 13 GUI assembly conditions

V5 from Dassault Systemshas been implemented for this purpose. The software is accessed through the application programming interface (API). Since the software has a lot of features and functions, only the transformations for the path planning and simulation workbench have been implemented because this opens up the possibility to calculate the assembly times and do collision checks on the robot paths. Besides the DELMIA model, also VRML models for a quick visualisation may be generated for potential virtual reality investigations and a model in LaTeX format for documentation. More domain models can be easily added.

As a second application a paintwork simulation, more specifically a coating thickness simulation, was chosen because it covers a high level of detail and is tightly connected to the product development. In this way a wide range of functional design verifications is illustrated with these two applications. For the necessary type of paintwork simulation it is important to create a ‘good’ meshed surface of the assembly to be painted. Only the paintwork related parts should be considered and the mesh should have different groups. All this conditions however can be achieved with model transformations which are not explained in detail here since they require a lot of knowledge about the CFD calculation process and the solver used. Nevertheless it is shown that the different surface meshes can be automatically generated as it can be seen in figure 16.

Starting from there, the coating thickness simulation was generated in cooperation with the Fraunhofer Institute for Manufacturing Engineering and Automation (IPA) in Stuttgart. On the IPA side the volume mesh, the definition of the boundary conditions for the simulation, the simulation and the result post-processing was done. Since an individual volume mesh for every static spray pattern is needed,

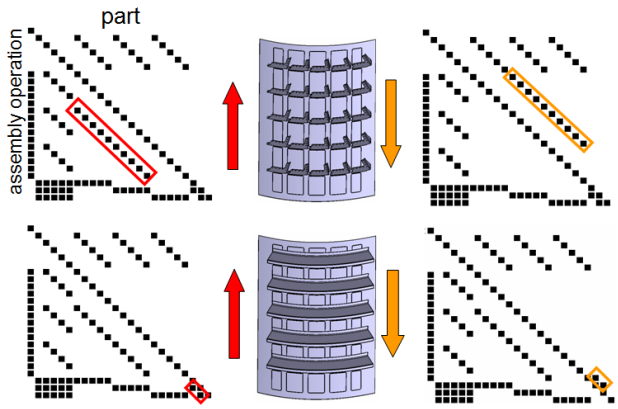

Figure 15 arbitary conditions

the complete automation of these activities as model transformations was realized. In this way the domain model transformations to DELMIA could be re-used and a simulation of the paint cabin and the robot paint paths could be easily generated (see figure 21) with varying boundary conditions.

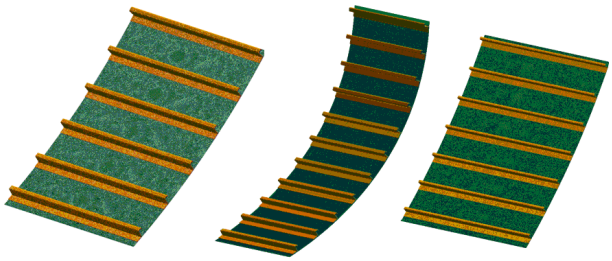

Figure 16 surface grid variants

5. RESULTS

Combining all the described approaches gives the ability to generate various variants of simulations very fast and without futher effort. All described components are generated automatically and are linked to each other. It is possible to change a parameter or the topology of the airplane panel and execute the whole process chain again. In every variant all resources are adapted for the actual configuration, all paths are consistent and domain models of the simulations can be generated. The DELMIA domain model can also check the pathes on collision. Therefore different scenarios can be quantified and compared with each other. Figure 17 shows 4 sequence variants and figure 18 shows 4 airplane panel variants. The paint simulation is done in the CFD simulation tool ANSYS-FLUENT (http://www.ansys.com/, 2012). The three-dimensional quasi-stationary turbulent flow in the coating was calculated including gravity, process air flow and the particle movements. Depending on the size of the airplane panel, there can be a lot of static spray patterns. In the shown configuration with 6 stringers 25 simulations are necessary. These static spray patterns can be superimposed to dynamic profiles with a program developed by the Fraunhofer Institute IPA (figure 20). Also the coating usage, coating thickness and the application efficiency will be calculated. Moreover, as already described, the path of the paint robot can be simulated using DELMIA to check on possible collisions (figure 21). The main aim of this work was to establish a complete process chain from the scratch to the digital factory. The main focus here was not to exactly model the simulations but with more effort it would be possible to generate models in industrial strength as it has alreday been successfully demonstrated in another industrial use case (Vogel et al., 2010). However the paint simulation serves as a simulation example for a very detailed functional verification.

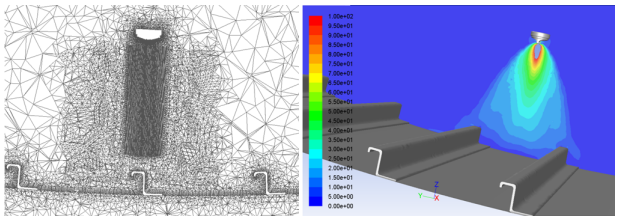

Figure 19 mesh for and velocity profile (courtesy Fraunhofer IPA)

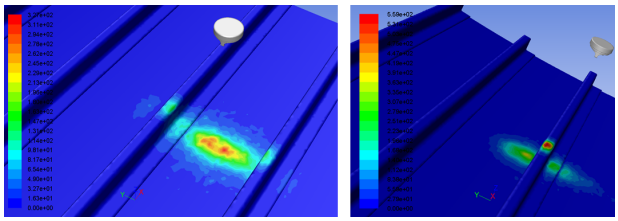

Figure 20 spray simulation (courtesy Fraunhofer IPA)

6. CONCLUSIONS

Summary. The use of graph-based design languages to generate models of the digital factory is shown. The whole process is illustrated with an airplane panel. Model transformations allow to build the central data model incrementally. The systematic storage of data in a central data model solves the consistency problem and is a possibility to exchange

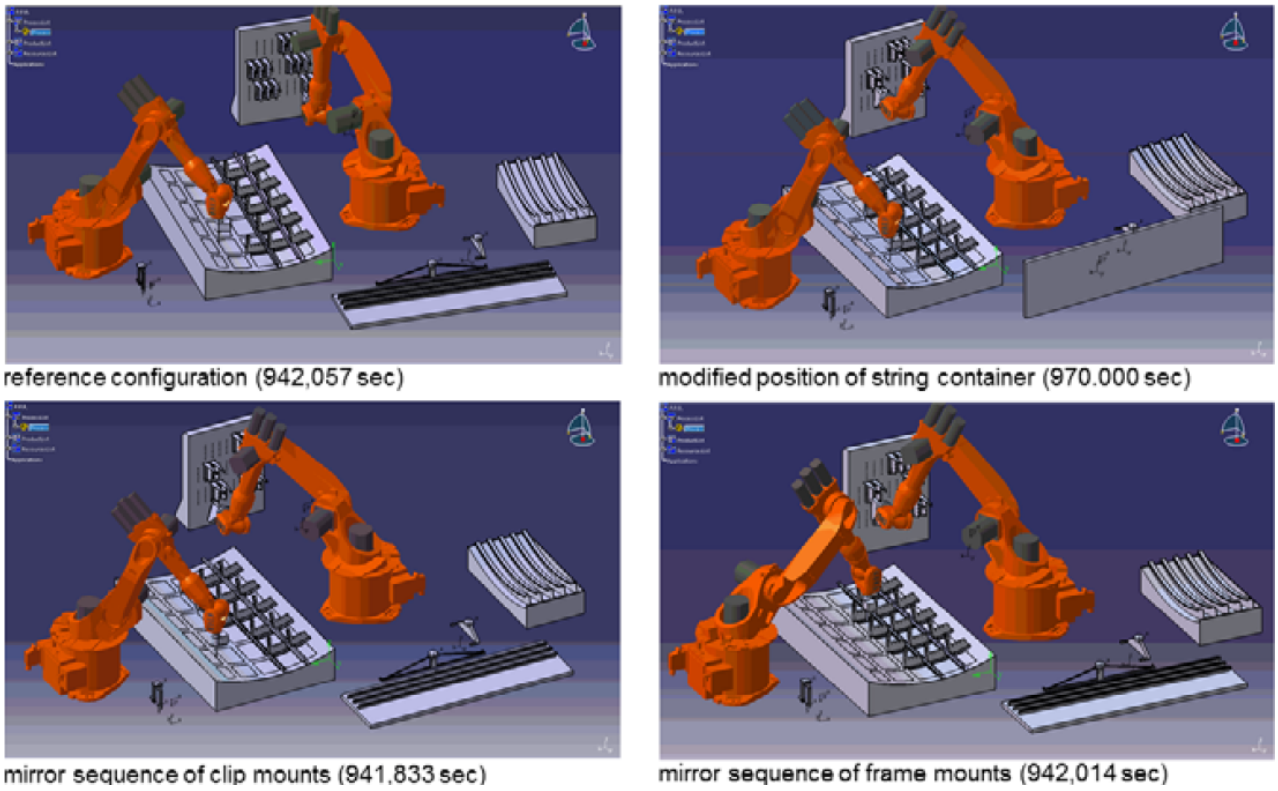

Figure 17 sequence variants

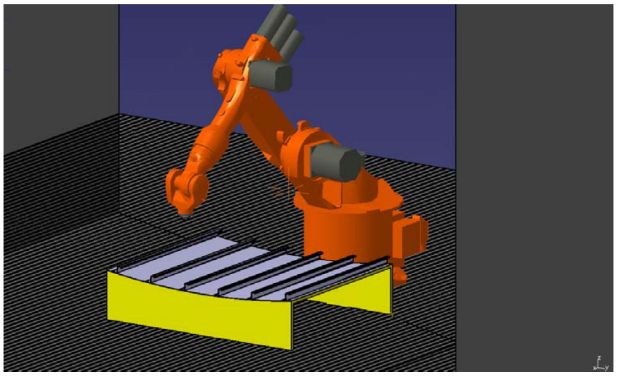

Figure 21 paint cabin simulation in Delmia

data in an unambiguous transformation. It is possible to automatically generate the different domain models. This is shown exemplary by a factory cell in DELMIA and paint simulation in FLUENT. Since the entire transformation process can be re-run, it is ensured that the centralized model is up to date and that it is possible to compare different variants without additional effort besides computation time. Currently the product, the layout configuration, the path planning, the assembly sequence and the resources can be varied. The advantages of the necessary assembly sequence algorithm are presented.

Also it has been clearly demonstrated that the digital data model created during the compilation process of graph-based design languages can successfully cover the whole engineering product life-cycle. This has been illustrated with application of design languages to aircraft panel design, painting simulation, digital factory process planning and simulation. The method allows to check which impact a product design change has on the models of the digital factory and to automate repetitive activities. The method is however not really useful when the process is only run once because of the substantial preliminary overhead, but in case of many iterative design loops as they are typical to engineering design processes, the method can be considered as very advantageous in both time (acceleration = shorter time to market) and money (less manual iterations = less cost).

According to the authors best knowledge this is the first time that the gap between product design and product manufacturing is digitally closed using graph-based design languages and the automated update of design changes and resulting manufacturing process updates was successfully demonstrated for both parametrical and topological design changes. All components like the variants in figure 4, the se-

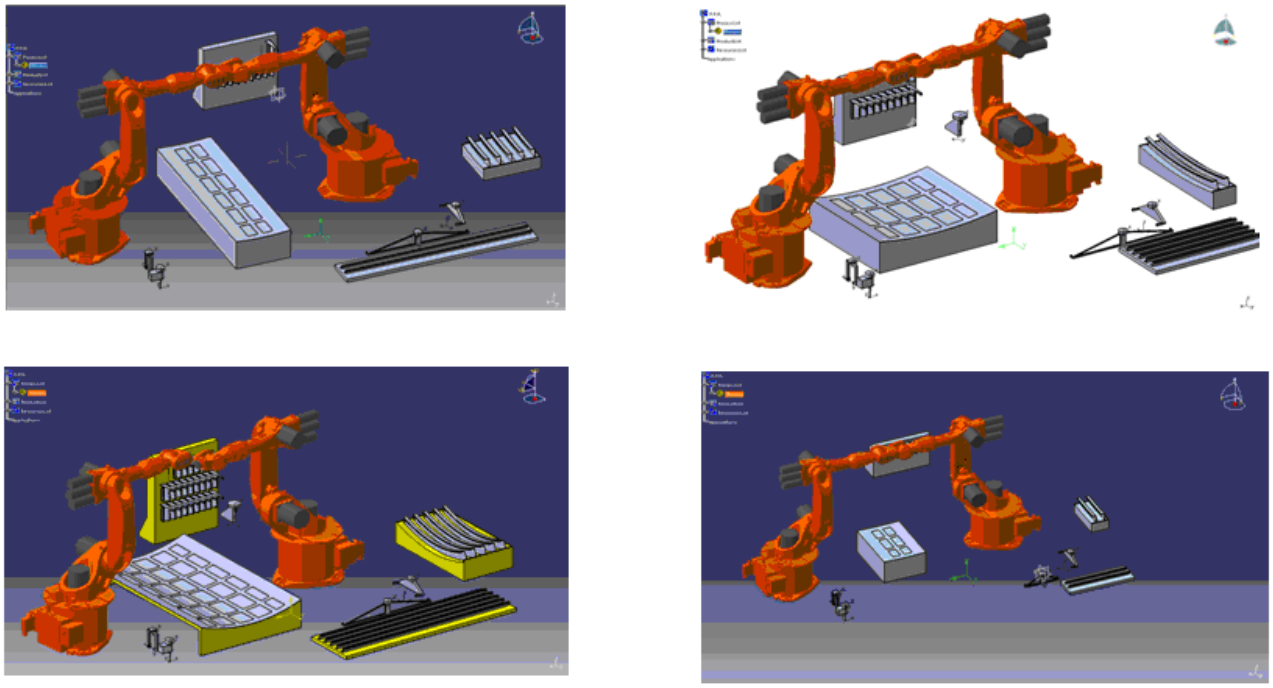

Figure 18 panel variants

lection of the robots in 10, the calculation and generation of the domain models for the assembly sequence in figure 17 and the domain model of the paint simulation are generated automatically. All data is always consistent and a large diversity of design and factory variants is possible.

Discussion. Due to the high upfront investment design languages are only profitable in industrial applications if the process chains are used many times (Vogel et al., 2010). For are few design and planning cycles this overhead may not be justified. However in case of extensive design optimazation loops or in the context of a product family the know-how re-use effect pays of. The whole process approach can also be applied to sub-processes.

The overhead could be significantly reduced if a 'factory-backbone' as a central datamodel already exists. Thus it has to be clarified if and how model transformations can be organized and held re-usable in large projects. Since similar approaches in computer science have the same problems, solutions might be adopted from there.

Outlook. A spray paint simulation was shown which simulates the behavior of paint droplets. Similarly, each step in the whole process could be represented with more detail and pushed to the actual level of the frontier of research. As another example, the path planning could be done at a higher semantic level of meaning as it is currently done in DELMIA where all path planning activity works with a sequence of 'semantically flat' coordinates. It is expected that reasoning about path planning and an appropriate conflict resoulution in a design language will be possible on a much higher semantic level, since the geometry of the design object is created with the very same language. Inside the design language, the geometry knowledge is therefore available at a much higher semantical level, thus hopefully opening up new possibilities for path planning and collision avoidance.

ACKNOWLEDGMENT

This work has been funded by the Baden-Württemberg Stiftung. For the results of the paint spray simulation, the authors thank Dr. Oliver Tiedje and Dr. Karl-Heinz Pully (both Fraunhofer IPA), for providing figures 19 and 20 as well as the fruitful cooperation and kind support in creating the simulation behind figure 21.

REFERENCES

Alber, R. & Rudolph, S. (2003). A Generic Approach for Engineering Design Grammars. Proceedings of the AAAI Spring Symposium "Computational Synthesis".

Bergholz, M. A. (2005). Objektorientierte Fabrikmodellierung. PhD thesis, TH Aachen.

Bley, H. & Franke, C. (2001). Integration von Produkt- und Produktionsmodell mit Hilfe der Digitalen Fabrik. wt Werkstatttechnik online 91.

Böhnke, D. (2009). Erstellung einer Flugzeugentwurfssprache für die aerodynamische Analyse mittels CFD-Methoden. Studienarbeit, Universität Stuttgart.

Booch et al. (1999). The Unified Modeling Language User Guide. Addison Wesley.

Cuiper, R. (2000). Durchgängige rechnergestützte Planung und Steuerung von automatisierten Montagevorgängen. PhD thesis, TU München.

Dombrowski et al. (2001). Visionen für die Digitale Fabrik. Zeitschrift für wirtschaftlichen Fabrikbetrieb (ZwF) 96, pp. 96–115.

Fuhr, J. (2010). Regelbasierte Generierung von Strukturkomponenten einer Flugzeugrumpfsektion mithilfe von Entwurfssprachen. Studienarbeit, Universität Stuttgart.

Gross, J. & Rudolph, S. (2011). Hierarchie von Entwurfsentscheidungen beim modellbasierten Entwurf komplexer Systeme. Tag des System Engineerings.

Grote, K.-H. & Feldhusen, J. (2007). DUBBEL - Taschenbuch für den Maschinenbau. Springer.

Haq, M. & Rudolph, S. (2004). EWS-Car: A Design Language for Conceptual Car Design. Proceedings of Numerical Analysis and Simulation in Vehicle Engineering, pp. 213–237.

http://www.ansys.com/ (2012). ANSYS Fluent. Website.

http://www.catia.com (2012). Catia V5R18. website.

http://www.delmia.com (2012). Delmia V5R18. website.

http://www.iils.de (2012). IILS mbH. website.

Irani, M. R. & Rudolph, S. (2003). Design Grammars for Conceptual Designs of Space Stations. Proceedings of International Astronautical Congress.

Jonas, C. (1999). Konzept einer durchgängigen, rechnergestützten Planung von Montageanlagen. PhD thesis, TU München.

Kapp, R. (2011). Eine betriebsbegleitendes fabriksimulationssystem zu durchgängigen unterstützung der kontinuierlichen fabrikadaption. PhD thesis, Universität Stuttgart.

Kormeier, T. & Rudolph, S. (2006). Topological Design Of Shell Structures By Design Languages. Proceedings of Design Engineering Technical Conferences.

Kuehn, W. (2006). Digital factory - integration of simulation from product and production planning towards operative control. European Conference on Modeling and Simulation.

OMG (2003). Model Driven Architecture.

OMG (2009). UML Superstructure Specification. 2.2.

OMG (2010). SysML Specification. 1.2.

Reichwein, A. (2011). Application-specific UML Profiles for Multdisciplinary Product Data Integration. PhD thesis, Universität Stuttgart.

Rudolph, S. (2002). Übertragung von Ähnlichkeitsbegriffen. Habilitationsschrift, Universität Stuttgart.

Rudolph, S. (2003). Aufbau und Einsatz von Entwurfssprachen für den Ingenieurentwurf. Forum Knowledge Based Engineering, CAT-PRO.

Schäfer, J. & Rudolph, S. (2004). Satellite design by design grammars. Aerospace Science and Technology, pp. 81–91.

Schuh et al. (2002). Integration als Grundlage der digitalen Fabrikplanung. VDI-Zeitschrift integrierte Produktion 144, pp. 48–51.

Serrano, D. (1987). Constraint management in conceptual design. PhD Thesis, Department of Mechanical Engineering, MIT, pp. 64–67.

Stachowiak, H. (1973). Allgemeine Modelltheorie. Springer.

VDI 3633-5 (2000). Simulation of systems in materials handling, logistics and production - Integration Of Simulation Into Operational Processes. VDI-Guidelines.

VDI 4499 (2008). Digital factory Fundamentals. VDI-Guidelines, pp.3̃.

Vogel, S. et al. (2010). Design and Development of Exhaust Aftertreatment Systems Based on a Graph-based Design Language. Heavy-Duty-, On- und Off-Highway-Motoren. Internationale MTZ-Fachtagung, 5.

Westkaemper et al. (2003). Digitale Fabrik nur was für die Großen? wt Werkstatttechnik online 93, pp. 22–26.

Whitney et al. (1999). Designing Assemblies. Research in Engineering Design, pp. 223.

Wiendahl, H.-P. (2002). Auf dem Weg zur Digitalen Fabrik. wt Werkstatttechnik online 92, 4, pp. 121.