Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 41 (2016) 141 – 146

www.elsevier.com/locate/procedia

48th CIRP Conference on MANUFACTURING SYSTEMS - CIRP CMS 2015

Approaching the Dilemma between Plan and Value in Computer Aided Engineering of Production Machines

Denis Özdemir*, Werner Herfs, Christian Brecher

Laboratory for Machine Tools and Production Engineering (WZL) of the RWTH Aachen University, Steinbachstr. 19, 52072 Aachen, Germany

* Corresponding author. Tel.: +49-241-80-27459; *E-mail address:* D.Oezdemir@wzl.rwth-aachen.de

Abstract

Today, manufacturers do not fully leverage the potential of Computer Aided Engineering (CAE) for the design of machine tools and production machines. Common design tools for drive systems and other machine components build on an algebraic system description. However, dynamic quality criteria that have a significant impact on the achievable accuracy and productivity cannot be evaluated with purely algebraic descriptions. Generally, dynamic criteria can be incorporated in signal-oriented models from control engineering, but the modeling process is time-consuming, knowledge-intensive and the reusability of models is limited. This paper presents a methodology to simplify the optimization of machine components, such as feed drives, while taking dynamic quality criteria into account. The method is based on the object- and component-oriented modeling language Modelica. For the quick development of models with limited expert-knowledge the models are integrated in an extensible model library. Modern mathematical optimization methods support the systematic search for a combination of system components and parameters with regards to the defined quality criteria. Therefore different design alternatives can quickly be analyzed and compared.

Peer-review under responsibility of the scientific committee of 48th CIRP Conference on MANUFACTURING SYSTEMS - CIRP CMS 2015

Keywords: Design of products, systems, families and platforms; Variety management techniques; Mathematical Optimization; Simulation

1. Introduction

During the last decades the application of tools for Computer Aided Engineering (CAE) for designing manufacturing systems, production plants and whole factories has steadily increased. Time savings in testing and start-up are the main motivation for CAE and simulation [1,2]. The aim is to identify weak spots during the design phase avoiding subsequent time-consuming and costly iterative improvements at the real manufacturing system.

A wide variety of CAE tools is currently available: starting from task-specific spreadsheet calculations and sizing tools, to static and dynamic simulation, up to co-simulation or model integrations that work with several software environments. The motivation for using more advanced CAE methods is typically that a deeper understanding of the system behavior can be gained by considering more interdependencies. This results in more testable requirements during the engineering phase and potentially less uncertainty regarding functionality. But with the level of detail, also the amount of required data, the modeling and simulation effort as well as the required expertise increases, see Fig. 1.

Regarding simulation, a recent survey among German manufacturers of machine tools shows that from the available methods only Finite Element Analysis (FEA) under static constraints is widely applied [3]. While most companies see high potential in virtual prototypes, few indicate that they use advanced methods such as co-simulation, systematic parameter optimization or multibody simulation with flexible bodies. The study identifies the lack of qualified employees and data as well as license costs as the main obstacles. Thus, there is a dilemma between the value of simulation results and the required planning efforts. CAE-tools that allow a reliable test of requirements and functionality with limited data and expertise at low efforts for modeling and simulation are generally not available.

This paper explores a concept of how such a CAE system can be built up for the system design stage. The main idea is to leverage object-orientation with model-libraries that enhance model-reusability. The new aspect is to augment requirements

Peer-review under responsibility of the scientific committee of 48th CIRP Conference on MANUFACTURING SYSTEMS - CIRP CMS 2015
doi:10.1016/j.procir.2016.01.017

to behavior models making it possible to employ mathematical optimization algorithms for finding a system configuration in a large search space that optimally fulfills given requirements.

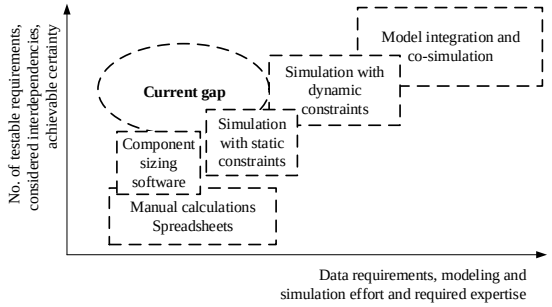

Fig. 1. Current gap in comparison with existing CAE approaches

2. Process for System Design of Production Machines

Typically, machine manufacturers offer different series that can be configured and adapted to customer requirements, see Fig. 2. This yields a combined process of variant configuration and customer-specific re-design [4]. Therefore engineering support is necessary in two phases: for series development to reduce the time expenditure on real prototypes and for customer adaption to meet specific requirements and to shorten the required time for start-up.

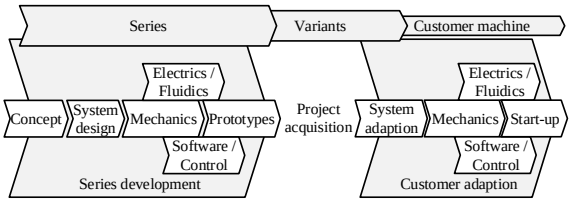

Fig. 2. Product development process for production machines based on [5]

Specifications for mechanical, electrical and software engineering are elaborated in the preceding stages of system design and adaption. Therefore, these stages have a significant impact on time and costs in subsequent phases. Especially the mechatronic interactions have to be considered, e.g. during feed drive design, to avoid later iterations.

So far system design for production machines is mostly based on textual descriptions complemented by schematic sketches and approximate calculations. Seldom decisions are validated by CAE tools due to limited data availability and high modeling efforts [6]. However, a systematic search in the solution space with regards to the requirements would be particularly beneficial in this early stage due to the powerful lever. Thus, the aim is to provide an environment that enables the specification of topology and behavior during system design as well as requirement testing and automatic optimization of system properties. At the same time low modeling efforts and easy parameterization are prerequisites for practical applicability.

Fig. 3 illustrates the proposed system design process. In the first step, the engineer models potential system variants which are composed of connected components, e.g. of a feed drive, see Fig 4. For building these system variants no expertise in modeling is necessary since the model structure resembles the physical structure. The components, which are obtained from a newly developed library, contain component-specific behavior descriptions. Therefore declarative rather than imperative modeling is used meaning that the engineer does not derive an explicit mathematical formulation, but specifies a topology. The overall system behavior is automatically derived from this topology. Moreover, the models have a granularity that allows a-priori parametrization with supplier data reducing the time for the search of data.

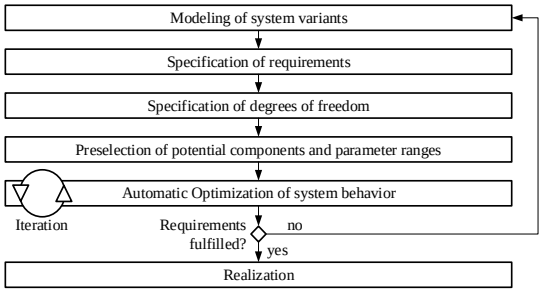

Fig. 3. Process model for system design

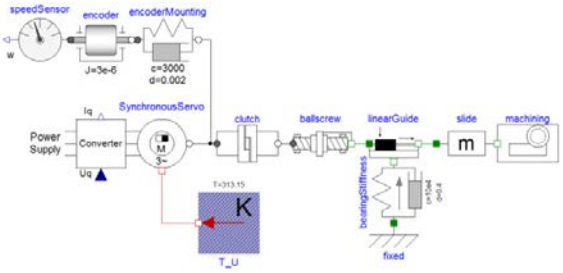

Fig. 4. Exemplary topology of a system variant for a feed drive

For system design the behavior has to be analyzed in context of the requirements. Behavioral requirements can correspond to limiting values, e.g. permissible torque, or to objectives, e.g. minimum cycle time. On the one hand, requirement verification necessitates the calculation of adequate metrics, e.g. a lifetime calculation. On the other hand, these metrics need to be compared to the requirements. Therefore, the components of the library comprise the necessary equations to calculate metrics as well as parameters (e.g. permissible values) for evaluation. Application-specific requirements are augmented to the system design in Step 2 of Fig. 3. Considering complex models this automatic requirement evaluation can significantly facilitate simulation analysis with regards to effort and expertise.

System optimization implies that optimal values for the degrees of freedom are determined with respect to the formal requirements. The degrees of freedom can either be single parameters or parameter sets corresponding to a supplier component, e.g. a servo motor. Some parameters or components are often a-priori set due to restrictions of configuration logic, mechanics, and electrics. The other parameters or components are regarded as the degrees of freedom of the optimization problem.

The search space for these degrees of freedom generally needs to be limited by excluding component or parameter ranges for avoiding long computation times.

During system optimization the model is executed iteratively with different values for the degrees of freedom leading to different levels of requirement fulfillment. The requirements are automatically evaluated in each step and the optimizer chooses new values taking the preceding evaluations into account. Automated result evaluation significantly reduces the time for cumbersome comparisons of different results. Moreover mathematical optimization allows reducing the total number of simulation runs in comparison to an evaluation of the whole search space. If the requirements cannot be fulfilled with any parameter configuration for any of the modeled system variants, the system design process has to start from the beginning again. Otherwise realization begins with the validated and optimized system design.

3. Environment for the System Design Process

The environment for the system design process was initially developed for feed drive sizing and has been introduced in a preceding paper, see Fig. 5. The primary objective of the environment is to support the system design process as described in Section 2. Therefore the environment is composed of three modules. The first module serves to model the system variants by means of a library with standard elements. Therefore the behavioral equations need to be encapsulated inside a component leading to the paradigm of object-orientation. Apart from considering the system behavior, the modeling and simulation environment has to allow the specification of design requirements. After simulation the evaluation results of the requirements are transferred to the optimizer in form of objectives and constraints. The optimizer chooses new parameters and/or components on the basis of the evaluation results. For component selection a database is connected to the optimizer that comprises available supplier data. Accordingly, behavior models and requirements are parametrized with the parameter sets of the database.

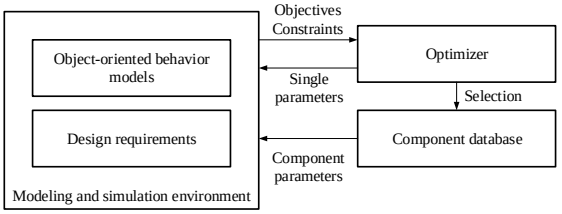

Fig. 5. Environment for the system design process [7]

The system design process involves different actors, see Fig. 6. The central actor is the engineer in charge of system design. Here, generally no expertise with regard to detailed model creation can be assumed, but it should be possible to use component models from a library and to define the system topology by linking components. The models are therefore developed in advance by domain and modeling experts, e.g. on side of the component supplier. Hence, the modeling experts are responsible for the extension of the model library and for the validity of the models. The modeling does not only include the actual behavior, but also the specification of component inherent requirements. In addition, it is important to ensure continuous data maintenance, which involves for example entering new component data from supplier catalogues and data updates.

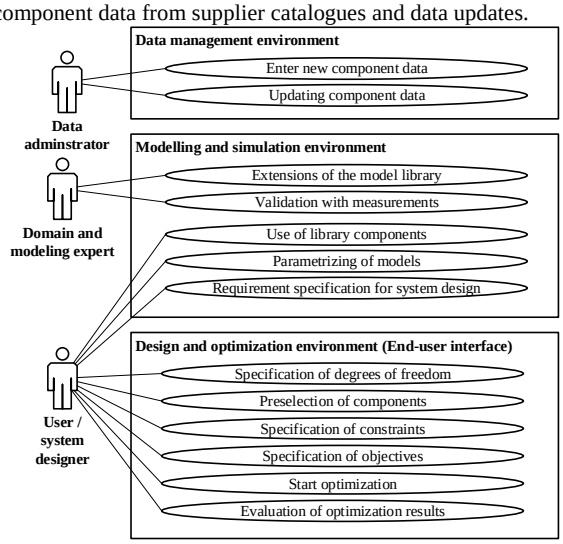

Fig. 6. Use case diagram for the design and optimization environment

The system designer interacts with two of the modules: with the modeling and simulation environment and with the optimization environment. After modeling a system variant with library components, the components have to be parameterized. The user specifies either individual parameters, such as the mass of the machining table, or component specific parameter sets. In addition the user needs to specify the load data, e.g. motion or force curves, as well as requirements concerning the system as a whole such as precision requirements. In contrast, requirements that arise from permissible data do not have to be explicitly modeled by the user, as these are included in the components of the library.

For automated solution search the design problem has to be formulated as a mathematical optimization problem. Since expertise in mathematical optimization generally cannot be assumed, a user interface has been developed that facilitates the formulation. In this context the user first defines the degrees of freedom in terms of parameters and components. The search for optimal parameters and components is speeded up and made more robust by limiting the search space and providing reasonable starting values. If multiple objective criteria are taken into account, it is required that the user weights the single objectives or that objectives are reformulated as boundary conditions, e.g. by choosing a limiting value for the cycle time instead of minimizing it. Alternatively the optimizer can search for Pareto fronts, from which appropriate solutions are determined a-posteriori.

The optimization result covers both: on the one hand, parameters and components that solve the design problem optimally; on the other hand, the resulting values for objectives and constraints. Since parts of the requirements do not relate to the behavior model, such as availability or price, it is helpful to know

not only the optimum design solution but also those alternatives that satisfy the most fundamental requirements. Therefore not only the values of the optimal solution are of interest, but also the individual solutions calculated on the way to the optimum. In any case, the user has to be informed to what extent the requirements are satisfied.

4. Object-Oriented Behavior Modeling

The proposed design process with the associated actors leads to requirements regarding the nature of modeling, the modeling language and the modeling and simulation environment. The primary objective is to reliably validate the key requirements while reducing the modeling effort.

There are generally two approaches to facilitate behavior modeling. Either potential configurations are predefined by modeling experts so that the system designer only choses a configuration and parametrizes it, or the designer builds up the topology himself by choosing, connecting and parameterizing existing components. In the first approach, the modeling language plays a minor role as long as an appropriate user interface is developed. While this approach is relatively independent of the modeling language, the latter necessitates that equations are encapsulated within components. This is accompanied by the demand for object-orientation, which allows this type of model reuse.

The object-oriented approach offers the advantage of being able to adapt the model topology to the actual application. In addition, object-orientation also allows predefining topologies. Other benefits are an intuitive understandability of the models and a high transparency regarding the underlying equations.

Therefore, an essential requirement for the modeling language and the simulation environment is to enable object-orientation and the development of model libraries with intuitively understandable graphical symbols. In addition efficient solution algorithms for the models are required since simulation time multiplies with the number of optimization iterations. The simulation environment has to provide an open interface that allows the exchange of information between simulation, optimizer and database.

In order to simulate object-oriented behavior models a model hierarchy is necessary: from the graphical model to a mathematical formulation, see Fig. 7. The top level comprises the library with behavior models. Elements of the library have connectors that describe potential interactions with other components or with the environment. The connectors are therefore the basis for building up the topology. Each component and connection in the editor has an underlying textual description that can be handled by the simulator. From the textual model descriptions a common system of differential-algebraic equations is generated that can be solved after bringing it to a state space form. Finally, the results can either be evaluated by the user or an automatic evaluation with respect to the requirements is executed.

In principle different languages can be used for building object-oriented models. In the context of mechatronics the Modelica modeling language [8] is considered as a de facto standard that is supported by different software companies [9]. The concept presented here is based on Modelica ensuring tool independence and making it possible to leverage a wide range of available model libraries.

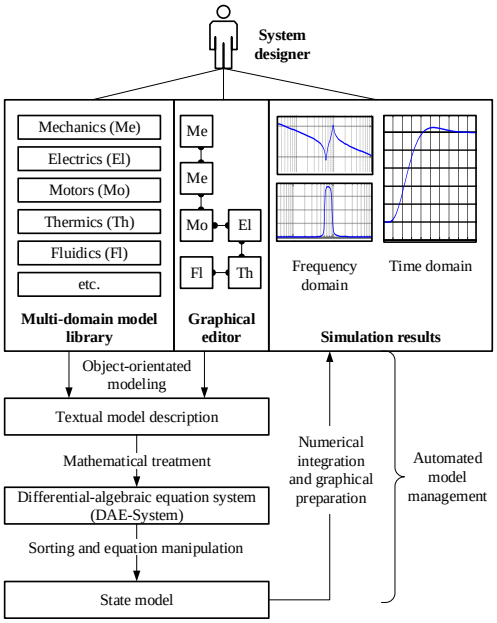

Fig. 7. Model hierarchy for object-oriented physical modeling based on [9]

5. Requirement Modeling and Behavior Metrics

Generally, simulation experiments alone give no indication how “good” a system design behaves. There are often many different variables that need to be observed, e.g. forces, times, vibration behavior and energy consumption. Therefore, requirement validation is time-consuming and prone to errors even for the experienced simulation engineer. The primary aim of requirements modeling is the effective and objective assessment of a design solution in terms of the simulated behavior. For this purpose, a metric is required that maps simulation results to scalar values making it possible to automatically evaluate simulation results and to leverage mathematical optimization methods.

The specification of requirements for production systems in the form optimization problems has been proposed already in in the early 1970s [10]. During the last decade the idea revived with the introduction of modern CAE tools [11]. A prerequisite for this approach is a good metric for the behavior. Typically metrics regarding system behavior are computed by p-norms:

$$\left\| m \right\|_p := \left(\sum_{j=1}^{n} \left| y_j - y_{ref,j} \right|^p \right)^{1/p} \tag{1}$$

where m defines the metric, y a system variable and y_{ref} the corresponding reference value. For some requirements, however, p-norms are not sufficient and the metrics need to be extended to generalized averages. Considering servo gearboxes

for example the effective output torque $M_{Therm,\,eff}$ with respect to gear heating is given by

$$M_{Therm,\,eff} = \sqrt[1,2]{\frac{|n_1|\cdot t_1 \cdot |M_1|^{1.2} + \ldots + |n_n|\cdot t_n \cdot |M_n|^{1.2}}{|n_1|\cdot t_1 + \ldots + |n_n|\cdot t_n}}, \tag{2}$$

i.e. absolute values of the torque $|M_1| \ldots |M_n|$ are weighted with the angle covered [12]. Generalizing this corresponds to the weighted power mean value

$$\|m\|_{p,w} := \left(\sum_{j=1}^{n} w_j \cdot y_j^p \right)^{1/p}, \tag{3}$$

with $y_j = |M_j|$ and $w_j = (|n_j| \cdot t_j) / \sum |n_j| \cdot t_j$. The metrics according to (1) and (3) can be calculated a-posteriori on the basis of the simulation results, but to avoid a second evaluation step the equations can by handled analogously to the other behavioral equations as an Lp-norm:

$$\|m\|_{p,\tilde{w}} = \sqrt[p]{\frac{\int_{t_0}^{t_{end}} y(t)^p \cdot \tilde{w}(t)\,.dt}{\int_{t_0}^{t_{end}} \tilde{w}(t)\,.dt}}, \tag{4}$$

where $\tilde{w}$ has been introduced to weigh $y(t)^p$. The example given with (2) can thus be written in an integral form by setting $y(t) = |M(t)|$ and $\tilde{w}(t) = |n(t)|$. In the simulation result, the metrics are calculated during simulation as the quotient of two state variables:

$$M_{Therm,\,eff} = \sqrt[1,2]{\frac{x_1}{x_2}}, \quad \text{with} \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} |n(t)| \cdot |M(t)|^{1,2} \\ |n(t)| \end{pmatrix}. \tag{5}$$

Within Modelica different blocks have been developed to calculate metrics according to (4). On this basis different kinds of engineering metrics are calculated, e.g. maxima of force or torque, mean energy consumption and bearing life. In the next step these metrics are formulated as requirements using objective functions *f* and inequality constraints *g*, i.e.

$$\begin{aligned} &\min \left\{ f_1(\mathbf{m}), f_2(\mathbf{m}), \ldots, f_k(\mathbf{m}) \right\}, \\ &g(\mathbf{m}) \le 0. \end{aligned} \tag{6}$$

To reduce the modeling effort behavior metrics (4) and requirements (6) are integrated into the behavior models of the library. For example the component model of the servo gearbox contains the equation for calculating the effective torque according to (5) and the requirement

$$g(\mathbf{m}) = M_{Therm,\,eff} - M_{Therm,\,perm} \le 0, \tag{7}$$

where $M_{Therm,\,perm}$ denotes the permissible torque. Here, the user does not need to make an explicit requirement specification, since (5) is part of the library model and $M_{Therm,\,perm}$ is obtained from the component database, see. Fig. 5.

One example for an objective function is average consumption of electrical power, i.e.

$$f_1(\mathbf{m}) = \frac{1}{t_{end} - t_0} \int_{t_0}^{t_{end}} U(t) \cdot I(t)\,.dt, \tag{8}$$

6. Optimization

Automatic evaluation of requirements is the basis for optimization, where an algorithm searches for parameters and components from the database that optimally fulfill the requirements (6). Therefore the modeling and simulation environment OpenModelica has been coupled to two different optimization packages (NOMAD [13] and DAKOTA [14]) making it possible to leverage a broad scope of optimization algorithms. As a prerequisite the optimization algorithms must be able to handle inequality constraints and categorical variables. Handling inequality constraints is necessary for considering requirements such as (7). Categorical variables are introduced for choosing components whose parameter sets are stored in the database. Gradients with respect to the degrees of freedom can only be obtained by perturbation since the simulation platform does not provide these directly. Therefore the advantage of fast convergence with gradient-based solvers cannot be leveraged. Fast and robust convergence with multiple constraints and categorical variables has been achieved with Mesh adaptive direct search as implemented in NOMAD, see [7]. Moreover, NOMAD allows to calculate one-dimensional Pareto fronts making it possible to solve bi-objective optimization problem a-posteriori. Multiple objectives as indicated in (6) are handled in the developed optimization environment by weighting.

7. Applications

One application is the sizing of feed drives such as Fig. 4, for details see [7]. In comparison to existing sizing tools for servomotors, the approach presented here allows not only to choose the right motor, but to solve the combinatory problem of optimizing the feed drive system as a whole, i.e. motor, controller parameters and mechanical transmission elements. Moreover it is possible to include the dynamical behavior, e.g. the lowest mechanical eigenfrequency, as a criteria for optimal design. While standard feed drive applications are covered with the developed library, experts can add components for specific scenarios.

The design of fluidic systems is an additional application scenario, see Fig. 8 for a simple exemplary system. Here, objectives can be cycle time and energy consumption, while the limiting values of cylinder and proportional valve have to be satisfied. Choice of pump, valve and cylinder form the degrees of freedom.

The approach can also be applied to other domains such as process chains planning, see Fig. 9. In this context it is helpful

that Modelica allows to simulate hybrid models, i.e. a combination of continuous and discrete-event equations. A potential application scenario is to optimize the parameters of inventory control under volatile boundary conditions.

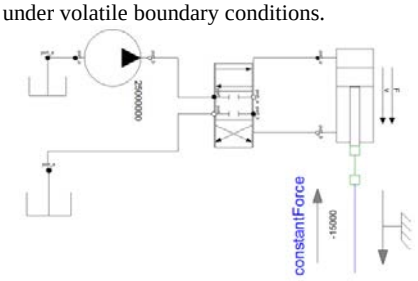

Fig. 8. Modelica model of a hydraulic cylinder with proportional valve

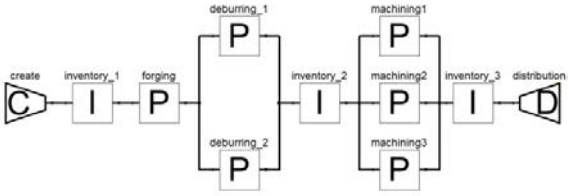

Fig. 9. Modelica model of a process chain

8. Conclusion

This paper proposed that the mathematical optimization of object-oriented behavior models with augmented metrics and requirements can reduce modeling efforts and required expertise. Thus, the dilemma between required planning effort and value gained from simulation is reduced. Regarding one functional unit of a production system the gain ΔG from simulation can be written as:

$$\Delta G = n \cdot \left(\Delta p + \Delta c_{saved} \right) - \Delta c_{sim} , \tag{9}$$

where n denotes how often the functional unit is applied, Δp is the price premium for improved functionality and shorter time-to-market, Δc_{saved} are the saved costs in engineering and start up and Δc_{sim} are the additional costs for applying simulation. Latter costs do not scale with the number of applications of the functional unit. Reducing Δc_{sim} can therefore make simulation more attractive for machine and plant engineering, where small batch sizes are common.

With current developments such as the common data standard ecl@ss and standard interfaces for simulation tools such as the Functional Mock-up Interface the design process can be further simplified, making simulation-based decision making more attractive.

Acknowledgement

The authors would like to thank the German research Foundation DFG for the kind support within the project “Optimierung des Systementwurfs von Maschinen und Anlagen auf Basis komponentenorientierter Verhaltensmodelle”.

References

[1] Altintas Y, Brecher C, Weck M, Witt S. Virtual machine tool. CIRP Annals-Manufacturing Technology 2005;54:115–38.
[2] Abdul Kadir A, Xu X, Hämmerle E. Virtual machine tools and virtual machining - A technological review. Robotics and Computer-Integrated Manufacturing 2011;27:494–508.
[3] Brecher C, Brockmann B, Daniels M, Wennemer M. Herausforderungen bei der messtechnischen Untersuchung von Werkzeugmaschinen. Zeitschr. f. wirtsch. Fabrikbetrieb 2014:885–8.
[4] Brecher C, Karlberger A, Herfs W. Synchronisation of Distributed Configuration Tools using Feature Models. In: ElMaraghy HA, editor. Enabling Manufacturing Competitiveness and Economic Sustainability: Proceedings of the 4th International Conference on Changeable, Agile, Reconfigurable and Virtual production (CARV2011), Montreal, Canada. Berlin: Springer; 2012, p. 339–345.
[5] Simon S. Benchmarking im Werkzeugmaschinenbau: Ein Beitrag zur wettbewerbsfähigen Produktentwicklung. Technische Universität Darmstadt: Dissertation; 2006.
[6] Komoto H, Masui K. Classification of design parameters with system modeling and simulation techniques. CIRP Annals - Manufacturing Technology 2014;63:193–6.
[7] Herfs W, Özdemir D, Lohse W, Brecher C. Design of Feed Drives with Object-Oriented Behavior Models. In: Troch I, Kugi A, Breitenecker F, editors. MATHMOD 2015 - 8th IFAC International Conference on Mathematical Modelling, Vienna; 2015, p. preprint.
[8] Modelica Association. Modelica - A unified object-oriented language for physical systems modeling: Language Specification - Version 3.2. [July 09, 2012]; Available from: https://modelica.org/documents/ModelicaSpec32.pdf.
[9] Janschek K. Mechatronic Systems Design: Methods, Models, Concepts. Berlin: Springer; 2012.
[10] Spur G. Optimierung des Fertigungssystems Werkzeugmaschine. München: Hanser; 1972.
[11] Yoshimura M. System design optimization for product manufacturing. Concurrent Engineering 2007;15:329–43.
[12] SEW Eurodrive. Servo Gear Units Catalog: Project Planning Information for Servo Gear Units. 1677241.
[13] Le Digabel S. Algorithm 909: NOMAD: Nonlinear optimization with the MADS algorithm. ACM Transactions on Mathematical Software (TOMS) 2011;37:44.
[14] Adams B, Dalbey K, Eldred M, Swiler L, Bohnhoff W, Eddy J et al. DAKOTA, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis: Version 5.2 User's Manual. Albuquerque: Sandia National Laboratories; 2011.