22nd INTERNATIONAL DEPENDENCY AND STRUCTURE MODELING CONFERENCE, DSM 2020

CAMBRIDGE, MASSACHUSETTS, USA, 13 – 15 October, 2020

A Role-Activity-Product Model to Simulate Distributed Design Processes

Ferdinand Wöhr[1,2], Simon Königs[1], Philipp Ring[1], Markus Zimmermann[2]

[1]BMW Group
[2]Technical University of Munich

Abstract: Computational models can be used to study product design processes. We propose an agent-based simulation model that incorporates, first, the role of design parties involved, second, their design activities to drive the product towards a design goal, and, third, a product model based on a so-called attribute dependency graph. A carefully tailored data model is presented for two roles, a system designer and a component designer, and their specific activities. The performance of the resulting development processes is measured by the time needed to reach the overall design goal. The model is applied to the design of an electric vehicle battery pack consisting of two components that are developed separately by a total of four parties. The results show plausible interaction between the design parties and how uncoordinated activity by one party can decrease process performance.

Keywords: design process, distributed design, process simulation

1 Introduction

Successful product development is essential for many companies to stay competitive and profitable. It can be characterized by a short development time, high product quality and low cost accumulation (Ulrich, 2016). As all of those three factors are closely linked to the design process and in particular to the order of design activities, rearranging, adding or removing certain design tasks can lead to a better product development performance. (Chanron, 2004). However, predicting the effects of modified design processes in advance is quite challenging, because computational models need to reflect the complexity and irregularity of real design processes. These real-life circumstances exist for several reasons. One major reason is the attempt to fulfill many, often times contradictory requirements linked to quantities of interest at the system level. These quantities of interest depend on multiple design variables, which need to be adjusted properly by the various design parties involved in order to accomplish all design goals (Zimmermann, 2017). Computational models of product design processes need to represent these functional interdependencies in detail. A common approach regarding this matter is to allocate design variables and quantities of interest to individual decision makers, who are part of a game-theoretic simulation (Lewis, 1997) or an agent-based model (Hulse, 2018). Many products are also being developed in a distributed environment, which further increases the complexity (Königs, 2016). This can lead to a limited communication between parties and an additional effort to integrate all subsystem solutions. If information between distributed design parties is not shared on a regular basis it also may happen that the design work of one party is based on an obsolete status of another parties design work. This effect has already been investigated by the use of an agent-based model (Wöhr, 2020).

Another reason for complexity, which has not been considered in many computational models yet, is the presence of different engineering roles. Especially since many products are being developed by multidisciplinary teams following the leadership of a systems engineer this might be an important factor. Hence, this paper investigates the impact of different engineering roles in distributed design processes by the use of an agent-based model in order to improve understanding and enhance simulation model accuracy. For this purpose a data model is presented and the behaviour of a System Designer is described mathematically. The application is shown for the design of an electric vehicle battery pack.

This paper is organized as follows: Chapter 2 reviews the current state of the art with respect to engineering roles in design processes. In Chapter 3 the design process model is introduced. The simulation method is explained in Chapter 4. Chapter 5 illustrates the technical details of an electric vehicle battery pack. Results of the study are presented in Chapter 6 followed by the Conclusion and Outlook, which are drawn in Chapter 7 and 8.

2 State of the Art

2.1 Engineering Roles

As products grow in size and complexity they are most likely being developed by a team, which consists of highly specialized engineers and professionals. An example for such a multidisciplinary team can be found in (Ulrich, 2016). Besides a typical *Team Leader* the authors suggest a *Mechanical Designer*, *Electronics Designer*, *Industrial Designer*, *Manufacturing Engineer*, *Purchasing Specialist* and *Marketing Professional*. Thus, every role in the team is responsible for a discipline-specific design goal, which needs to be accomplished. A list of roles, which are normally used in systems engineering is provided by (Sheard, 1996). Among others the authors suggest a *System Designer*, *Requirement Owner*, *System Analyst*, *Technical Manager* and *Coordinator*. Due to the holistic character of systems engineering those roles are more general and tend to represent product design from a different, rather superficial, perspective. A very detailed description of a *Systems Engineer* can be found in (NASA, 2007). According to the authors this role coordinates, monitors and directs all activities of a systems engineering team in order to ensure that the technical requirements of a system are fully satisfied. This involves evaluating design trade-offs, balancing technical risks and allocating requirements.

In summary, two small findings can be presented. First, there is no uniform description or standardized documentation of engineering roles. This is most likely due to the fact that every engineering project has unique characteristics and therefore individual role preferences. Second, a distinction between designers who operate on the system level (e.g. Systems Engineer) and others who operate on the component level (e.g. Mechanical Designer) can be made. The interaction between those roles is subject of the present study.

2.2 Modeling Design Processes

In general, many modeling techniques can be used to analyze design processes, like System Dynamics or Network Theory. Each of them provides individual strengths depending on the scope of investigation. Matrix-based methods, like the Design Structure Matrix or Multiple Domain Matrix, can be used to study complex design processes as well (Eppinger, 2012). Especially in case of many interrelated process elements (components, people, activities) they provide a powerful tool to document and visualize dependencies in order to improve transparency and understanding. Furthermore, rearranging certain rows and columns or clustering nearby matrix entries might lead to a more efficient design process in terms of development time or product quality. In order to cover the various forms of complexity which are mentioned above, however, agent-based models are most promising. Especially since they allow modeling of human behaviour on a micro level and simultaneously evaluating the process performance on a macro level. Many aspects of design processes, like solidarity (Canbaz, 2014) or learning (Hulse, 2018) have already been studied by the use of such models. Engineering roles, however, have not been part of any noticeable research and, thus, shall be examined in this paper.

3 Design Process Model

In order to decompose and analyze distributed design processes, which include roles, a formalized representation of all relevant objects and their relation is required. This can be achieved by establishing a data model of the design process according to the Unified Modeling Language Notation, see (Weilkiens, 2008). Such a data model (Class Diagram) is displayed in Figure 1. It serves as an orientation for the following chapters of the paper.

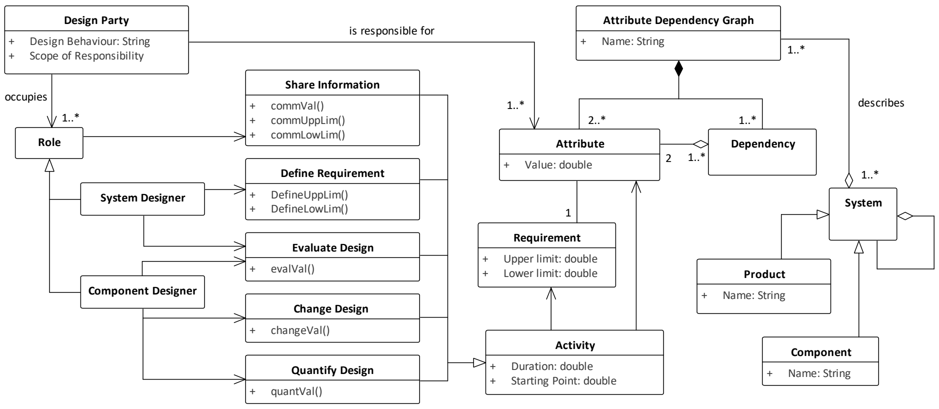


Figure 1. Data (Role-Activity-Product) Model of the Design Process

An essential part of the model is the representation of a system, like a product or a component, which can include multiple subsystems. The technical characteristics of such a system can be described by a certain number of attribute dependency graphs, see (Zimmermann, 2017). According to the notation of the formalism attributes, like design variables or quantities of interest, are displayed as vertices while functional dependencies between them are displayed as directed edges. Such a graph (see Figure 2) is hierarchically organized and allows the distinction between different product design levels. In case of a standard design scenario those levels can be called component level (bottom), subsystem level (middle) and system level (top). Design scenarios, which include more complexity and, thus, more attributes as well as dependencies, may contain multiple subsystem levels between the bottom and top. Every attribute has a value and is linked to a requirement with an upper and lower limit. In total there are five activities, which interrelate with those three numerical quantities (attribute value, upper and lower requirement) as follows:

- **Change Design (CD)**: increases or decreases the value of an attribute. This is supposed to represent the work of an engineer, who tries to improve his component performance by adjusting a design variable. From an industrial perspective this could be changing a CAD-file or modifying a part drawing. Different models of how engineers may behave during this task are presented in (Gurnani, 2008; Wöhr, 2020).
- **Quantify Design (QD)**: determines the value of an attribute based on the values of all other attributes which are connected to this element of the graph through a functional dependency. In comparison to real design processes this could be the analysis of some product performance by the use of a computational model (e.g. finite-element simulation), prototype experiment (hardware) or an expert assessment.
- **Evaluate Design (ED)**: examines whether the value of an attribute lies between the upper and lower limit of its requirement. This represents an engineer, who evaluates to what extend the component performance he is responsible for has already reached the associated design goal. In reality this occurs if simulations have been conducted and the results need to be checked with respect to whether or not they are satisfying.
- **Define Requirement (DR)**: changes the upper and lower limit of a requirement. This symbolizes the dynamic variation of requirements while technical trade-offs between subsystems are balanced and risks are managed. Most often this activity is carried out by engineers, who oversee the system performance and lead the process. A theoretical model of how engineers may behave during this task is presented in the Chapter 4.2.
- **Share Information (SI)**: communicates the value of an attribute and the limits of its requirement to other design parties. Addressing the importance of information exchange in distributed design processes, this activity represents the interaction between departments, teams and individuals. In reality information exchange may happen as a result of a personal conversation, team meeting or a data / file exchange.

However, not all of those activities are executed by the same engineering role. In this paper we assume that a *Component Designer* carries out certain design activities at the component (CD) and subsystem level (QD, ED and SI) while a *System Designer* executes certain design tasks at the subsystem (QD, ED, SI and DR) and system level (QD and ED). Every engineering role is occupied by a party (e.g. individual, team or department), whose design behaviour can be described mathematically. Each of the parties is linked to a certain number of attributes, which in sum define the scope of responsibility of the corresponding party. Each attribute within a graph must be exactly in one parties scope of responsibility.

4 Simulation

4.1 Agent-based Model

While the data model presented above focuses on representation and understanding, the computation of such a socio-technical system requires an additional simulation routine. In this paper we use a simplified agent-based model in order to represent and simulate the distributed decision makers, who try to solve a complex design problem in collaboration.

Yet, the model is rule-based, deterministic and quantitative. Due to simplicity learning of the agents is not included. Human design behaviour is described by mathematical formulae. An illustration of the model including three design parties is shown in Figure 2.

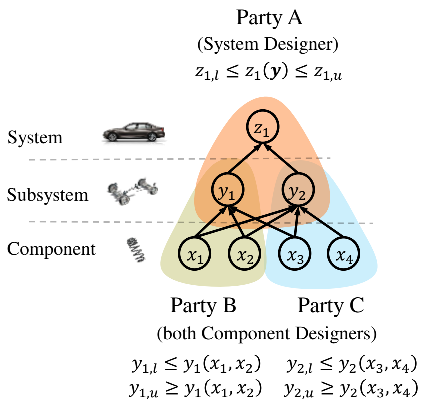

Party A
(System Designer)
$z_{1,l} \leq z_1(\boldsymbol{y}) \leq z_{1,u}$
System
Subsystem
Component
z_1
y_1
y_2
x_1
x_2
x_3
x_4
Party B
Party C
(both Component Designers)
$y_{1,l} \leq y_1(x_1, x_2)$
$y_{1,u} \geq y_1(x_1, x_2)$
$y_{2,l} \leq y_2(x_3, x_4)$
$y_{2,u} \geq y_2(x_3, x_4)$

Figure 2. Agent-based model including an attribute dependency graph

Due to simplicity we assume that all of the activities a party is responsible for can only be executed in combination. Hence, we call this a design step. In general all parties included by the model try to reach their design goals by adjusting certain design variables they are responsible for or by formulating requirements to the next lower level (Equation 1). Furthermore some of the parties have to consider certain constraints as well (Equation 2).

$$z_{i,l} \leq z_i(\boldsymbol{y}) \leq z_{i,u} \quad \& \quad y_{i,l} \leq y_i(\boldsymbol{x}) \leq y_{i,u} \tag{1}$$

$$\text{Subject to: } g_j(\boldsymbol{x}) \leq 0 \quad j = (1, \ldots, m) \tag{2}$$

Since a Component Designer has to select a combination of values for his design variables every time a design step is being carried out, a theoretical model for the corresponding design behaviour is required. Most of the common models, which originate from the field of game theory, assume rational or bounded rational behaviour, see (Lewis, 1997; Gurnani 2008). A gradient-based model, however, includes the time parties need to find a local solution (Wöhr, 2020). While this effect can be observed in reality as well the model is used for the present study. A different assumption regarding the human design behaviour could impact the results significantly, however, this will not be studied. In case of a multi-objective design scenario (not displayed in Figure 2) the model needs to be combined with an optimization strategy. In this paper a so-called *worst case scheme* is used (Equation 3):

$$max \left\{ \frac{y_i - y_{i,l}}{y_{i,l}}, \frac{y_i - y_{i,u}}{y_{i,u}}, \frac{y_{i+1} - y_{i+1,l}}{y_{i,l}}, \ldots \right\} \tag{3}$$

Hence, the combination of values a Component Designer selects every time a design step is being carried out depends on the distance between the performance of the current design and the requirements, which have not been reached yet. The number of design steps every party is executing over the course of a design process is labelled as n (e.g. n_A for Party A).

4.2 Behaviour of a System Designer

Besides describing the behaviour of a Component Designer, it is necessary to develop an additional model, which defines how a System Designer derives requirements every time a design step is being carried out (see: **Define Requirement**). In particular the dynamic change of subsystem requirements depending on the solution offered by the Component Designers needs to be described mathematically. Such a model is shown in Figure 3 below.

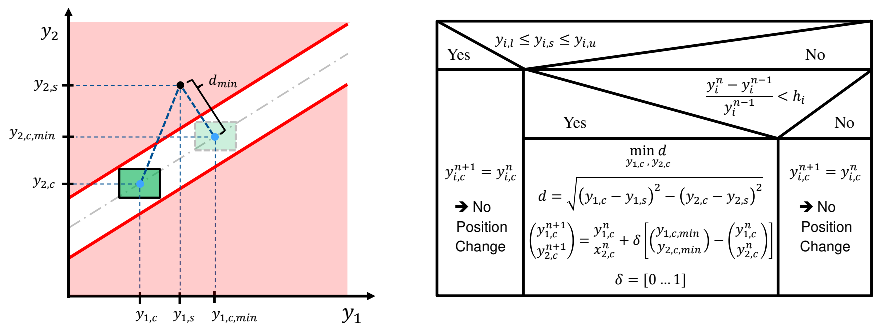

y_2
$y_{2,s}$
$y_{2,c,min}$
$y_{2,c}$
d_{min}
$y_{1,c}$
$y_{1,s}$
$y_{1,c,min}$
y_1
$y_{i,l} \le y_{i,s} \le y_{i,u}$
Yes
No
$\frac{y_i^n - y_i^{n-1}}{y_i^{n-1}} < h_i$
Yes
No
$y_{i,c}^{n+1} = y_{i,c}^n$
➔ No Position Change
$\min_{y_{1,c}, y_{2,c}} d$
$d = \sqrt{(y_{1,c} - y_{1,s})^2 - (y_{2,c} - y_{2,s})^2}$
$\begin{pmatrix} y_{1,c}^{n+1} \\ y_{2,c}^{n+1} \end{pmatrix} = \frac{y_{1,c}^n}{x_{2,c}^n} + \delta \left[\begin{pmatrix} y_{1,c,min} \\ y_{2,c,min} \end{pmatrix} - \begin{pmatrix} y_{1,c}^n \\ y_{2,c}^n \end{pmatrix} \right]$
$\delta = [0 \ldots 1]$
$y_{i,c}^{n+1} = y_{i,c}^n$
➔ No Position Change

Figure 3. Distance-based behaviour of a System Designer

On the left side of Figure 3 a simple design scenario is displayed, which supports the explanation of the model. While both red areas do not include any satisfying design, the white area in between represents the complete solution space. Each system variable (y_1 and y_2) is defined by a Component Designer, who tries to fulfill the associated design goal. The boundaries of the green box to the left side (center: $y_{1,c}$, $y_{2,c}$) symbolize the corresponding upper and lower requirement at the start. Both, the position and shape of the box, are controlled by the System Designer. Assume the solution of both Component Designers ($y_{1,s}$, $y_{2,s}$) has converged (criteria: h_i) and is not located inside of the given box due to constraints that may have been reached. As a result, the System Designer has to adjust the upper and lower requirement of both system variables in order to facilitate the design work of both Component Designers. According to the model this is achieved by moving the center of the box to the position within the complete solution space, which is the closest to the current subsystem solution ($y_{1,c,min}$, $y_{2,c,min}$). If the upper or lower limit of a system variable is reached, the box is located right at the border. An adjustment of the box-shape is not performed. However, the model includes a parameter, which determines how far the box is moved towards the optimal position within the complete solution space (δ). This can be interpreted as the caution of a System Designer, who has to change multiple requirements at once while operating in a highly complex engineering environment. As a description for the human behaviour we call it "*distance-based-model*."

5 Industrial Example

5.1 Technical Description

Electromobility causes major difficulties for car manufacturers due to the fact that many unknown technologies have to be applied in combination. In particular the design of an electric vehicle battery pack is challenging since two highly interdependent systems need to be fitted into a tight assembly space. In this paper a simplified model of the battery pack, shown in Figure 4, is used to simulate and analyze different design process scenarios.

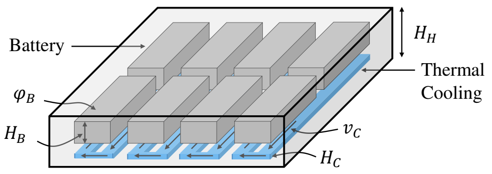


Figure 4. Model of an Electric Vehicle Battery Pack

Inside of the housing (H_H), which serves as a geometrical constraint, multiple batteries (height: H_B; power density: φ_B) are joint together to provide electrical power (P_E). Since they are generating heat ($\dot{Q}_B$) a thermal cooling system (height: H_C; fluid velocity: v_C) is installed as well. However, to ensure a sufficient heat flux leaving the system ($\dot{Q}_C$) the cooling requires a certain mechanical power (P_M) too. The system power (P_{sys}), which is necessary for the vehicle propulsion and drive, can be determined as follows (Equation 4):

$$P_{sys} = P_E - P_M \qquad (4)$$

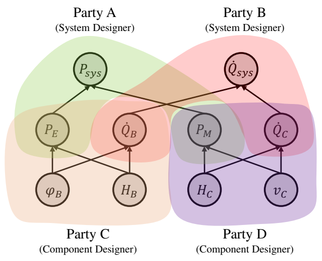
In this paper, we define a requirement for the system power ($150kW \leq P_{sys} \leq 250kW$), which is comparable to reality and does not change over the course of the design process. The remaining heat flux ($\dot{Q}_{sys}$) entering or leaving the system is defined as (Equation 5):

$$\dot{Q}_{sys} = \dot{Q}_B - \dot{Q}_C \qquad (5)$$

To prevent thermal overheating or efficiency losses due to a cold battery, we define a requirement for the remaining heat flux ($-5kW \leq \dot{Q}_{sys} \leq 5kW$), which shall be constant. The electrical power and heat flux every battery is generating is related to its volume (V_B) and power density ($P_E \sim \varphi_B V_B$; $\dot{Q}_B \sim \dot{q}_B V_B$) as we assume that the heat generation rate ($\dot{q}_B$) is linear proportional to the power density. On the other hand, the mechanical power and heat flux of the thermal cooling system is related to the volume flow rate ($\dot{V}_C$), pressure drop (Δp_C) and temperature gradient (ΔT_C) inside of the channel ($P_M \sim \Delta p_C \dot{V}_C$; $\dot{Q}_C \sim \Delta T_C \dot{V}_C$), while the pressure drop is a function of the fluid velocity (v_C) and volume flow rate ($\dot{V}_C$).

5.2 Distribution of Engineering Roles

In alignment with reality we assume that multiple design parties are involved in the design process of an electric vehicle battery pack. The distribution of engineering roles and allocation of design variables & quantities of interest to different design parties is shown in Figure 5. At the top level of the attribute dependency graph (left side of Figure 5) two different design parties are responsible for the system power and the remaining heat flux (Part A and Party B). As System Designers they both try to accomplish their design goals by formulating subsystem requirements. The battery pack and the layout of the thermal cooling system is being designed by two other parties (Component Designers), who use their design variables at the bottom level to reach their design goals (Party C and Party D).



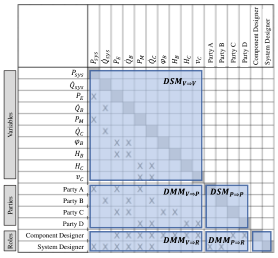


Figure 5. Distribution of roles / variables in an attribute dependency graph (left) and MDM (right)

All of the design variables can only be varied within a certain interval. On the right side of Figure 5 the socio-technical system is shown as a Multiple Domain Matrix, which provides a comprehensive documentation of all relevant elements and their relations. This improves the understanding for the complex dependencies between design parties, engineering roles and design variables as their interaction becomes heavily unintelligible.

6 Results

6.1 Simulation Setup

Now, the data model (Chapter 3) and simulation method (Chapter 4) is applied to the industrial example (Chapter 5). In order to study different design process scenarios, the number of design steps carried out by the four parties is being varied. Due to simplicity, we assume that the number of design steps carried out by Party A and B as well as C and D have to match $(n_A = n_B; n_C = n_D)$ All other factors are constant $(\delta = 1; h = 0.3)$. The numerical step size of both Component Designers and their initial guess at the start of the design process can be seen as a realistic assumption. Every simulation run has one hundred discrete time steps. All design steps of the four parties are equally distributed over time.

6.2 Detailed Analysis

First, a single design process is analyzed in detail in order to gain an understanding for the complex mechanisms behind. The evolution of system variables (blue markers) and requirements (green boxes) for $(n_A = n_B = 3; n_C = n_D = 25)$ is displayed in Figure 6 below.

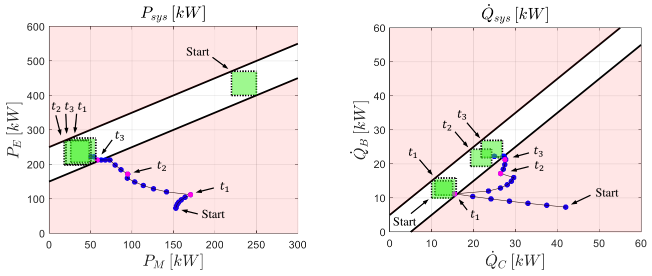


Figure 6. Evolution of system variables and requirements for $(n_A = n_B = 3; n_C = n_D = 25)$

In both graphics the moments of requirement changes are marked with t_1, t_2 and t_3. It turns out, that the initial requirements lead to a solution for the remaining heat flux, which almost lies in the specified box (t_1). However, the design goal for the system power is not reached at all. Thus, both System Designers, especially Party A, adapt their requirements. This facilitates the finding of a better solution for the system power, however, it also moves the current design further away from the required remaining heat flux (t_2). After another adjustment of the requirements, the design of Party C and D converges towards a satisfying solution for the system power and remaining heat flux (t_3). In total, it takes them eighty-five time steps to reach the specified boxes. This illustrates how complex and dynamic a distributed design process can become, if multiple engineering roles interact. However, this only represents a single design process, without the intention to generalize.

6.3 Parameter Study

For a more comprehensive analysis of the dynamics a parameter study is carried out. In this study the number of design steps carried out by Party A and B as well as C and D is varied within a certain interval. The process performance is assessed by the number of time steps both Component Designers need to find a solution, which lies inside of the both boxes (t). The results of the study, presented as contour lines, are shown in Figure 7 below.

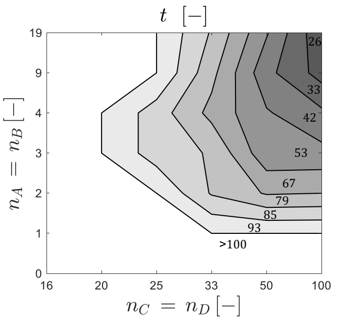


Figure 7. Development time depending on the number of design steps with $n_A = n_B$ and $n_C = n_D$

Both factors, which have been varied, show a strong coupling as their interdependency forms multiple Pareto frontiers. These lines reveal, that in some cases more requirement changes carried out by the System Designers lead to a lower process performance. Furthermore, they indicate, that at least one requirement change has to be carried out, so that both Component Designers can find a satisfying solution in time. The best process performance is reached if all design parties involved carry out a high number of design steps. Although these results provide quantitative information, they only represent a single design scenario under the assumption of multiple simplifications. Numerical optimization techniques as used in multidisciplinary design optimization cannot improve the results further, unless there are stricter requirements or targets.

7 Conclusion

In this paper a data model was introduced, which includes multiple engineering roles and their associated design activities. Combined with an agent-based model this allows the computational investigation of design processes with different roles. The application to the design process of an electric vehicle battery pack revealed, that both, the number of design steps carried out by the Component Designers and the number of design steps carried out by the System Designers impact the product development time.

8 Outlook

Future work will focus on the integration of additional engineering roles and the empirical validation. For this purpose a multi-actor experiment will be carried out under controlled conditions and the observed human behaviour may be used in order to calibrate the simulation on a micro scale. Furthermore, a study will be directed towards the question, whether the design process can be improved by clustering the MDM shown in Figure 5.

References

Canbaz, D., Yannou, B., Yvars, P. A., 2014. Resolving Design Conflicts and Evaluating Solidarity in Distributed Design. IEEE Transactions on Systems, Man, and Cybernetics: Systems, Vol. 44, No. 8

Chanron, V. , Lewis, K., 2004. Convergence and Stability in Distributed Design of Large Systems. Proceedings of DETC 2004

Eppinger, S., Browning, T., 2012. Design Structure Matrix Methods and Applications. MIT Press, Cambridge

Gurnani, A., Lewis, K., 2008. Collaborative, Decentralized Engineering at the Edge of Rationality. Journal of Mechanical Design, Vol. 130

Hulse, D., Tumer, K., Hoyle, C., Tumer, I., 2018. Modeling multidisciplinary design with multiagent learning. Artificial Intelligence for Engineering Design, Analysis and Manufacturing 1-15, https://doi.org/10.1017/S0890060418000161

Königs, S. , Zimmermann, M., 2016. Resolving Conflicts of Goals in Complex Design Processes – Application to the Design of Engine Mount Systems. Proceedings of the 7th International Munich Chassis Symposium

Lewis, K., Mistree, F., 1997. Modeling Interactions in Multidisciplinary Design: A Game Theoretic Approach. AIAA Journal, Vol 35, No 8

NASA, 2007. NASA Systems Engineering Handbook. URL: https://www.nasa.gov/sites/default/ files/atoms/files/nasa_systems_engineering_handbook.pdf (visited on the 14.04.2020)

Sheard, S.A., 1996. Twelve Systems Engineering Roles. Proceedings of the INCOSE Sixth Annual International Symposium

Ulrich, K.T. , Eppinger, S.D., 2016. Product Development and Design. McGraw-Hill Education. ISBN: 978-0-07-802906-6

Wöhr, F., Königs, S., Stanglmeier, M., Zimmermann, M., 2020. Simulation of Gradient-Based Individual Design Behaviour in Distributed Development Processes. Proceedings of the DESIGN Conference 2020

Weilkiens, T., 2008. Systems Engineering with SysML/UML. Modeling, Analysis, Design. dpunkt. Verlage GmbH, Heidelberg, Germany, ISBN: 978-0-12-374274-2

Zimmermann, M., Königs, S., Niemeyer, C., Fender, J., Zeherbauer, C., Vitale, R., Wahle, M., 2017. On the design of large systems subject to uncertainty. Journal of Engineering Design, 28:4, 233-254

Contact: F. Wöhr, BMW Group, Department of Functional Design, Knorrstraße 147, 80788 Munich, Germany, Phone: +49 170 1801486, ferdinand.woehr@tum.de