INTERNATIONAL DESIGN CONFERENCE – DESIGN 2020
https://doi.org/10.1017/dsd.2020.323

AUTOMATED PRODUCT FUNCTIONALITY AND DESIGN OPTIMIZATION INSTANCING A PRODUCT-SERVICE SYSTEM

P. Wolniak ✉, B. Sauthoff, D. Kloock-Schreiber and R. Lachmayer

Leibniz Universität Hannover, Germany
✉ wolniak@ipeg.uni-hannover.de

Abstract

When using product-service systems as a business model, new product development challenges and opportunities arise. Due to the possibility of customizing the product fleet depending on the user-scenarios, more product variants are possible and often necessary. Therefore, this paper presents an approach for the automated functionality and design optimization for user scenario specific use cases. The approach combines an optimization framework with a functional simulation model and a generative design approach CAD model. This results in a robust and simultaneously flexible design environment.

Keywords: design optimisation, generative design approach, knowledge-based engineering (KBE), product-service systems (PSS)

1. Introduction

The development in a product-service system (PSS) differs significantly from the typical business-to-customer or business-to-business model. Due to the shift of delivering the value proposition rather than the product, the requirements for the own products are not as strictly dedicated by the customer as in the general product development. This gives the developer a higher degree of freedom in the development of the product for meeting the requirements of the value proposition. However, it also imposes a more dynamical and flexible necessary reaction to e.g. market situation changes or new customer needs.

The distinction between the general product development process and the product development embedded in a PSS development is necessary for the consideration of the influence of the later phases of the product lifecycle as well as the internal dependencies between product and service parts (Meier et al., 2010; Schreiber et al., 2018). In general, PSS can be classified in the categories product-oriented, use-oriented and result-oriented (Tukker, 2004). While the product ownership changes from the producing company to the customer after the manufacturing and the following sale of the product, the ownership of the product stays with the manufacturing company within a use- or result-oriented PSS. Since the delivery of a value proposition (regardless whether it is achieved by product or service parts) is the main business goal of a PSS, not only the manufacturing but also the maintenance or further supply of consumable goods is of great importance (Thomas et al., 2008). Therefore, the requirements defined by the customer are shifting from the product to requirements specific for the value proposition itself.

This shift towards delivering solely the value proposition in a specified quality and cost gives the PSS company further freedom of designing the product and especially the structure of the entire fleet of products according to the service qualities. Hence, an even larger variety of product variants is

possible and may even be necessary to fulfill the business case. For this reason, a detailed investigation of the user scenarios and the subsequent possible alterations of the product on a cost-to-quality basis becomes mandatory.
Therefore, the scope of this paper is to propose an approach that allows computer-aided qualification of a high number of automatically generated variants within customer-specific scenarios. Finally, this qualification leads to an optimal functional fulfillment while enhancing the product fleet and lowering the cost for the PSS company itself.
The described approach is presented on the example of a PSS with the business case of a self-developed coffee machine for the automated delivery of coffee for high throughput. The described optimization approach supports the PSS company in the late conceptual phase of this coffee machine development for a user scenario-specific design and dimensioning of the product fleet according to the business case and the customers.

2. State of the art on automated design

To perform a computational synthesis of any kind a certain form of knowledge implementation is necessary. In a broader context, Alan Turing already distinguished between bottom-up and top-down approaches of computational knowledge implementation (Copeland, 2000). Characteristic for bottom-up approaches is a great amount of data used to perform training cycles in which the computational system patterns through dependencies and interrelationships and "learns" from scratch. Those approaches use e.g. neural networks, pattern recognition or classification, which are mathematical representations of complex functions. The training itself is an optimization of hyperparameters, leading to a certain level of goodness-of-fit of this function (Du and Swamy, 2019).. These approaches show progress in many fields of engineering and development within specific tasks. The use inside the conceptual phase of a functional assembly of a product development, on the other hand, is not yet included.
The top-down approaches are characterized as approaches where the knowledge about the dependencies and interrelationships is used at the top, thus being independent of the low-level details of the implementing mechanism. The idea behind these approaches is within the support for the solution space exploration, by implementing and using formal and informal knowledge. These approaches can be summed up as expert systems or knowledge-based engineering (KBE) approaches (Milton, 2007).
For the modeling of the solution space in these KBE approaches, the depiction of domain knowledge in the form of a declarative representation plays a vital role (Gembarski et al., 2016; Mescheder and Sallach, 2012) (Figure 1).

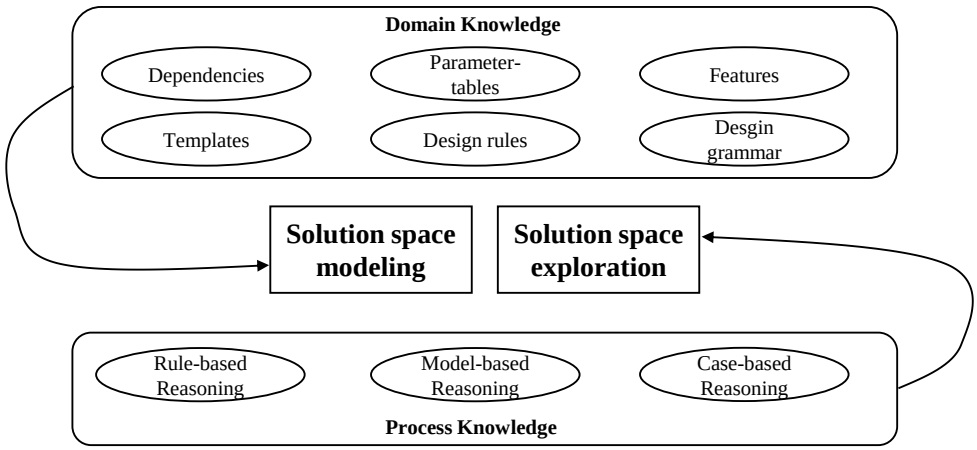

Figure 1. Solution space modeling and exploration (Gembarski et al., 2016)

Using this domain knowledge, problem-specific tasks can be implemented and solved. For a complete solution space exploration, inference knowledge and process knowledge is necessary. While the inference knowledge is the implementation of a single reasoning step depicted from the domain knowledge, the process knowledge puts these reasoning results in the context of the development task. For the implementation of these types of knowledge several methods exist (Schreiber et al., 1993).

https://doi.org/10.1017/dsd.2020.323 Published online by Cambridge University Press

A common way of implementing knowledge into geometric models is the parametric modeling. Ranging from the definition of implicit conditions through logical dependencies (combining standardized design elements with changing sizes) to more complicated constraint problems. Many parametric CAD-systems have modules for the application of so-called "design rules", which are used for the modeling of extensive relational networks of the components inside a CAD-system itself.
On this basis, the reasoning inside these KBE-systems can be implemented in a rule-based, model-based or case-based form. In a rule-based reasoning context, explicitly formulated rules are used to implement inference knowledge (Sabin and Weigel, 1998). With an increasing number of rules, the adaptation of the rule network becomes more complicated, as the implementation of a new rule includes a necessary consistency test. A pure rule-based system additionally lacks a comprehensive representation of the solution space, as every rule and therefore the majority of combinations have to be implemented in advance.
The model-based reasoning combines the implementation of models of knowledge, instancing e.g. analytical or numerical analysis features. The main difference is the implicit modeling of rules, which are represented through the model dependencies itself, adjusting the possible solution in every step depending on the boundary conditions as well as the admitted outcome. Representations of this form are logic-, constraint- and resource-based approaches (Sabin and Weigel, 1998; La Rocca and van Tooren, 2010; Hvam et al., 2008).
Case-based reasoning depends on a high storage of already known and determined solutions, categorized depending on its necessary information. By calculating the similarity of the new and the already known solutions, a measure of the usefulness of the stored solution can be given (Sabin and Weigel, 1998). Still, the solution space is limited, since the only outcome is the combination of already known solutions.
To overcome the disadvantages of the pure parametric modeling and the knowledge-based implementation, the so-called generative design approach (GDA) as presented by Sauthoff et al. (2016) yields a promising modeling strategy. This approach is based on the combination of a parametric geometry model and the generative aspect of automatically exchanging parts of the model, while still maintaining a continuously coherent model. The advantage of this approach in contrast to the pure generative approach is the parametric usability of the model after its creation in combination with the interoperability inside an optimization (Sauthoff et al., 2016; Li and Lachmayer, 2018). A further description of the functionality of the GDA is presented in section 3.3.

3. Research approach

3.1. Process description

Product development of physical assets, as known from the literature, has a distinct separation of processual phases. What many of the proposed processes, originating mainly in Europe, have in common is a serial sequence of these phases, namely a creative principle solution finding, a conceptual phase, a detailing phase and later on the elaboration and documentation (Pahl et al., 2007; Hubka, 1976; Koller, 1994). Another common point is the often highly iterative process of analyzing the so far determined product attributes and synthesizing the properties of a certain domain area. For a possible reduction of these often cumbersome iterations, a further computer-assisted or even automated development is used. Regarding this, every phase of the development process has its benefits and disadvantages for an automation.
The computational implementation of the creative phase of a solution finding is challenging, mainly because of the high amount of laborious analysis cycles. Furthermore, the synthesis, as well as the evaluation are complicated and interconnected processes leading to an enormous modeling effort. In contrast, with the support of CAD and CAE tools, the detailing phase is a computationally highly evolved application (Bryant Arnold et al., 2008). Largely detailed models are used for the verification of the conceptually implemented solutions. Configuration tools, as well as knowledge-based systems, are highly effective and commonly used approaches within this phase of the development process. However, changes on necessary alterations beyond a configuration process lead again to high modelling efforts, making this development phase unsuitable for automated conceptual changes.

https://doi.org/10.1017/dsd.2020.323 Published online by Cambridge University Press

Accordingly, the advanced concept phase yields an optimal application for an automated optimization process involving customer- and market-specific usage scenarios. Further, the opportunity of a substantial response to changing market situations or customer-specific needs is given, exceeding the possibilities of commonly used configuration tools in the detailed phase of development.
On this basis, the use of the presented approach is intended when a first set of solution principles is determined, remaining with further conceptual decisions like the actual optimal dimensions of the components or possible positioning inside an assembly.
As a result of the state of the art on automated design, the process presented in Figure 2 is proposed. Here the GDA is used as a basis for the geometric variant qualification due to the aforementioned disadvantages of existing approaches in the area of knowledge implementation. For a comprehensive concept evaluation the change in dimensional sizes has to be linked with the resulting functional effects. Therefore, a second functional model is used. To interlink these models and ensure a correct parameter transfer an optimization framework is used as the leading instance. The optimization algorithm inside the framework changes the parameters inside the restricted design space. The sequence of parameter change, as well as the model update, is determined in the optimization algorithm. While the algorithm itself simply determines the values of the design parameters, these values are passed on to the respective functional analysis and geometric model by a subsequent script with the implemented model logic. Every further synthesizing and simulation step is within those models and thus completely separated from the optimization process, with the advantage of a simpler and more transparent modeling.

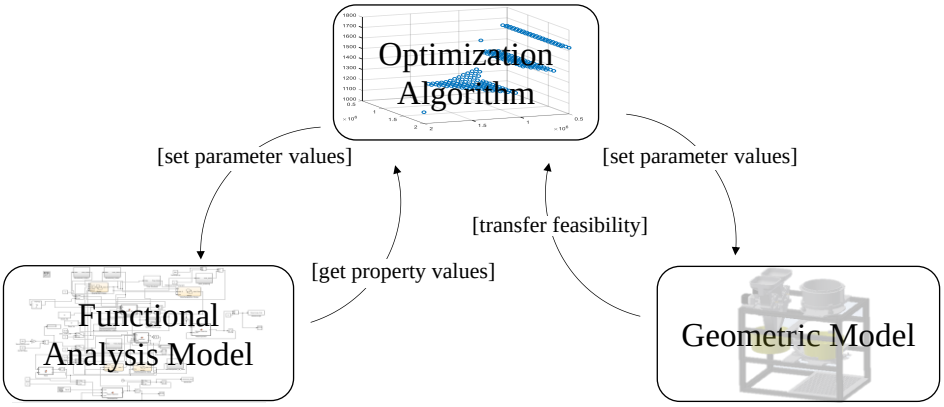

Figure 2. Process description

The process of parameter transfer is determined by the order of the model execution. First, every geometrical parameter is transferred to the geometric model. Here, the overall size, as well as positioning restrictions and the component prioritizations determine whether the parameter values given from the optimization algorithm lead to a solvable constraint problem and therefore to a feasible geometric solution. Using a simple death penalty constraint in the optimization algorithm, every unfeasible solution is excluded from further analysis. If a geometrically feasible solution is determined from the geometric model, the optimization algorithm transfers the parameter values to the functional analysis model, where the values of the actual objective functions are determined.

3.2. Optimization process

While the optimization of dimensions or performance of the components leads to a pure parameterization problem, another goal of this paper is to show the possibility of additional exchange of certain component types, extending the accessible solution space. Therefore, a differentiation has to be made between components that can be changed continuously in their size or form and components which are changed in discrete steps. While this is a highly individual distinction, general assumptions are possible. Self-developed as well as easily and with low-cost manufacturable components can mostly be seen as continuously dimensionable components. The cost of change is rather low or because of an in-house manufacturing possibility not of high significance. In the presented example of the coffee machine this refers to the tanks, the outer framework and several inner components like the motor shaft for the dosing unit.

https://doi.org/10.1017/dsd.2020.323 Published online by Cambridge University Press

Highly standardized or widely used components from the shelf are seen as discretely changeable components. Thereby, a first case base has to be created with the necessary parameters firmly defined and hence not changeable by the optimization algorithm. In this example, this refers to components like pumps, motors, electronics, and standardized pipe diameters.
While the optimization algorithm alters the actual geometrical values of the continuously changeable components, the discrete changeable components can be exchanged by assigning a parameter range to every individual component, standardized from zero to one. If the optimization algorithm determines a value in this parameter range, the component is automatically exchanged.

3.3. Geometric model

As described in section 2, the GDA is used to build-up a flexible model that is able to adjust to the changing parameter values, as well as to a possible exchange of parts or even a completely different positioning of several parts. Figure 3 gives an overview of the coffee machine model as designed in the GDA.

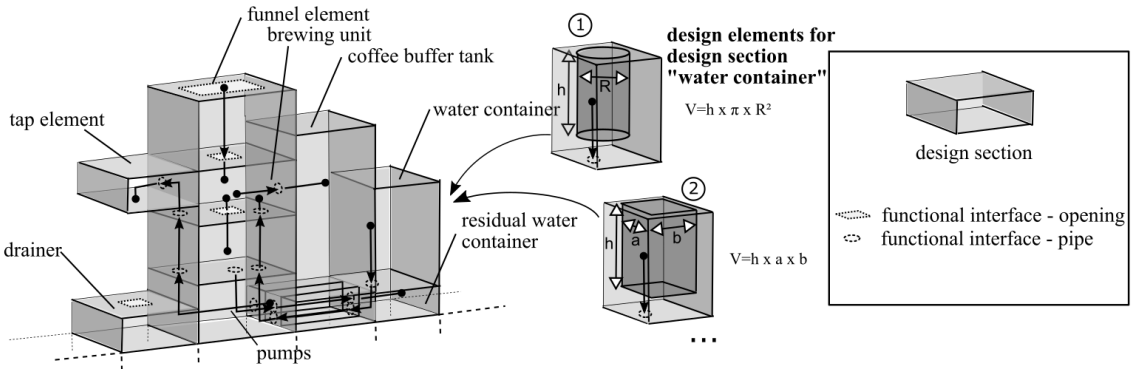

Figure 3. Design sections for the coffee machine

The overall structure of the product is built using design sections as a placeholder for any component and its actual design. The relative position of every design section is determined by the so-called design skeleton, while every design section is made from several virtual planes, restricting the space within. The design skeleton is modeled in a CAD assembly file with the virtual planes as references for the design elements, which then can be imported into the assembly file at the respective place and with the respective association to other parts. The right-hand side of Figure 3 depicts the design elements, in this example the water container, placeable in the design sections. Every design element is represented by a single CAD part file, containing the necessary design properties as well as knowledge through implementable design rules. As can be seen in the design element numbered as 1, the parameters for a cylindrical water tank (R and h), as well as the positioning in relation to the pipe interface is implemented in the design element itself, lowering the number of dependencies in the geometric model.
For a first assumption of the geometrical possibilities of the overall product, a spare geometry in the form of a simple rectangle can be used. For this, the volume of the design element has to be calculated and accordingly the size of the spare element is adjusted. Using rectangles for every design element simplifies the initial modeling and gives a first assumption of the volume occupied by every component. On this basis, boundaries for the overall size of the product can be checked.
For a robust and valid model build-up, regardless of the parameter combination at hand, a relative parameterization of the design skeleton is used. As can be seen in the lower part of Figure 4, the distances $l_1 - l_3$ and the overall length l are given. The actual parametric change of these distances is performed by adding a parameter like p_1 and inserting an equation as a design rule. By normalizing and restricting the parameter range of p_1, collisions or intersections of design sections can be avoided.
The functional interfaces, as a means of exchanging connected geometries, are also controlled by relative parameters. The upper part of Figure 4 gives an example of the functional interface for the opening of the funnel element, exchanging the coffee for the dosing unit through this opening. Again, the parameter p_2 is used to change the positioning and adjust it to the used design element. By connecting and referencing these parameters, a solvable dependency network is built, leading to a

https://doi.org/10.1017/dsd.2020.323 Published online by Cambridge University Press

robust parametric model. Additionally, various skeletons with different component positions can be used to enlarge the variety and the solution space.

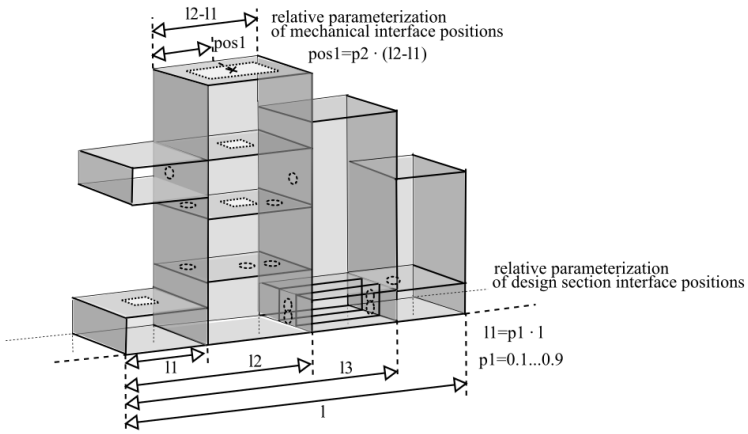

Figure 4. Relative parameterized interface positions

3.4. Functional analysis model

For an effective way of system modeling the separation of domain and process knowledge plays a vital role. Therefore, the approach in this article is to model the components of a given product as separate objects, inheriting their specific domain knowledge, while combining these objects to form the overall system (Wolniak et al., 2018, 2019). Figure 5 gives an overview of the overall functional analysis model structure.

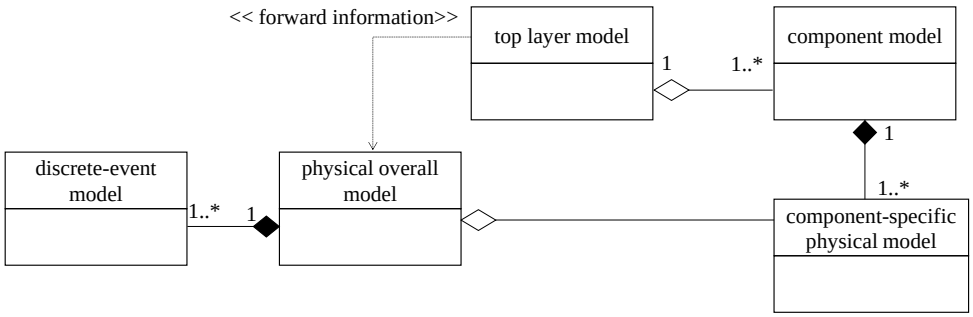

Figure 5. Functional analysis model structure

Thereby, every component of the actual product is represented as one object with its specific inputs and outputs, representing the information flow through the system. All these component blocks are inserted into the top layer model containing the logical connections between the component blocks. On this level, in a first step, the connections and dependencies between the components of the system are modeled, which is necessary to ensure a correct representation of the product. The previously mentioned exchange of discrete changeable components can be performed on this level. For this, each component created in the case base is loaded into the model. The currently active representative block is connected to the surrounding activated blocks. When changing the active block the status changes to suppressed, while activating the new block. A script, implemented into the optimization routine, analyzes the in- and outport identifiers and replaces the connections to the new linked blocks. Thereby, high model stability, as well as versatility, is guaranteed.

While this top layer model contains the connections and logical dependencies between the components and parameters, it is not the instance for the physical analysis of the system. Therefore, the actual physical analysis model of the system has to be built and connected to the top layer model.

Every component itself has a link to the respective specific physical model, necessary to simulate its outcome. As an example, the heating element, according to its design, e.g. as a flow heater or as a boiling tank, has a specific heating and flow curve as its outcome. This curve is, later on, necessary to

https://doi.org/10.1017/dsd.2020.323 Published online by Cambridge University Press

simulate the overall physical behavior of the system. According to the currently selected configuration in the top layer model, the necessary information is loaded into the physical overall model (Figure 5). Here, every component-specific physical model is combined and embedded in the system simulation. As in this case, the goal is to perform a functional analysis and, later on, an optimization of the system, the analysis model is a time-dependent discrete-event model. Discrete-event models as presented by Zeigler (2000) provide a common basis for system models which are described at an abstraction level where the time base is continuous, but during a bounded time-span only a finite number of relevant events occur. These events can cause the state of the system to change. In between events, the state of the system does not change.

An example of the implementation of this discrete-event model is shown in Figure 6 realized as a Stateflow model in the software Matlab/Simulink. This example shows the states, represented by the rounded blocks, the transitions, represented by the arrows and the transition conditions, represented by the parenthesized parameters and their distinct values. In this actual example, the transition between the states of the filling level of the buffer tank is modeled.

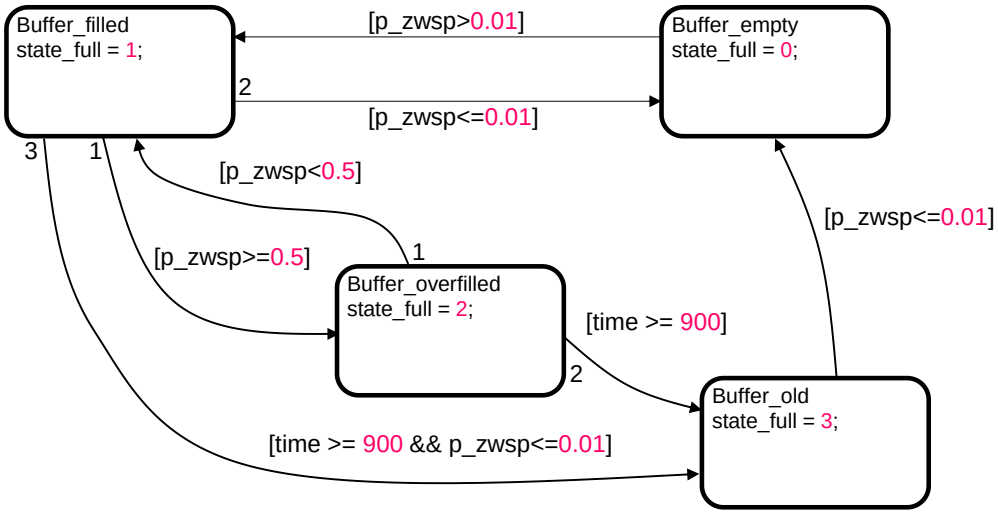

Figure 6. Stateflow example

The filling level can separated be in four different states: empty ($p_{ft} < 0.01$); filled ($0.01 < p_{ft} < 0.5$); overfilled ($p_{ft} > 0.5$); old ($0.01 < p_{ft} < 1 \,\&\&\, time \geq 900$). The parameter p_{ft} represents the normalized filling level from zero to one. Every time-step of the execution the transition conditions are checked and when fulfilled a transition is triggered. The model allows the system to be in only one state at each time-step. Furthermore, the connection order of the states allows a transition only from a connected state to another, allowing for a comprehensive constraint problem to be modeled graphically. The actual state of the system in this particular Stateflow model can be transferred to the overlying system, consisting of several more Stateflow models, triggering new events. Therefore, a complex system of conditions is constructible.

Implementing several instances for every discrete-event necessary for the system simulation into the physical simulation model yields the possibility of a holistic time and state-dependent system simulation.

4. Use case study

Using the previously described optimization process as well as the modeling techniques, an actual optimization of the described product of a coffee machine in a PSS is performed. Figure 7 depicts the activities throughout the usage of the coffee machine, leading to the disposal of the coffee. The functional analysis model and the geometric model are build-up as described in the previous section. The main changed parameters are in regard to the brewing chamber and the buffer tank, as they pose parameters with a high sensivity. Discretely changed parameters are the heating elements and the pumps. While the functional analysis model performs the analysis as shown in Figure 7, the geometric model is adjusted and checks for a solvable constraint problem and a feasible geometric solution.

https://doi.org/10.1017/dsd.2020.323 Published online by Cambridge University Press

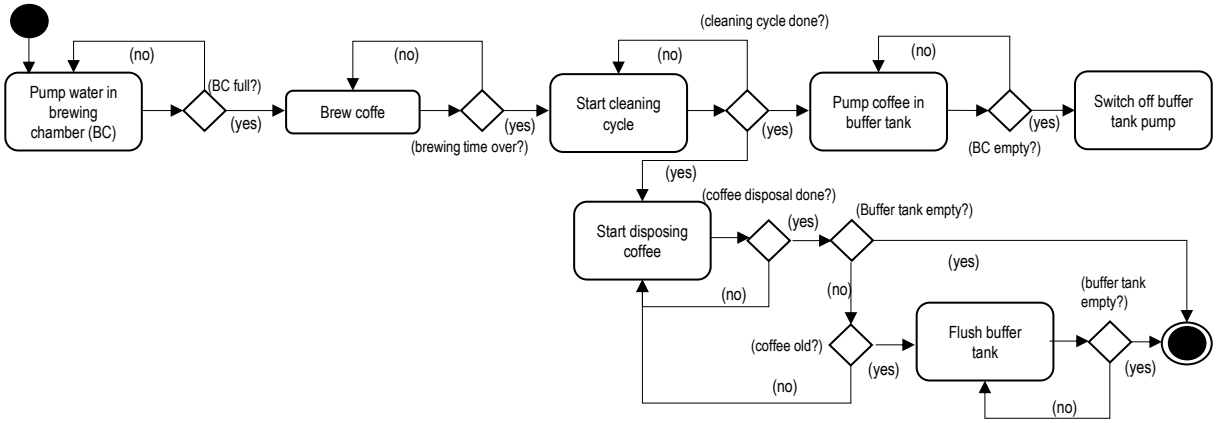

Figure 7. Activity diagram of the coffee machine usage

Two different scenarios are used for the optimization, as shown in Figure 8 (a), with the first one representing a hotel in the period of three hours during breakfast and the second one representing an office during the same three hours in the morning, when most employees are coming in to work. The restriction to three hours of a full day is explained by using representative times of the day with the highest utilization of the machine and therefore regarding the worst case for the design. The diagram of usage shows the number of cups being ordered at several time peaks of these three hours.

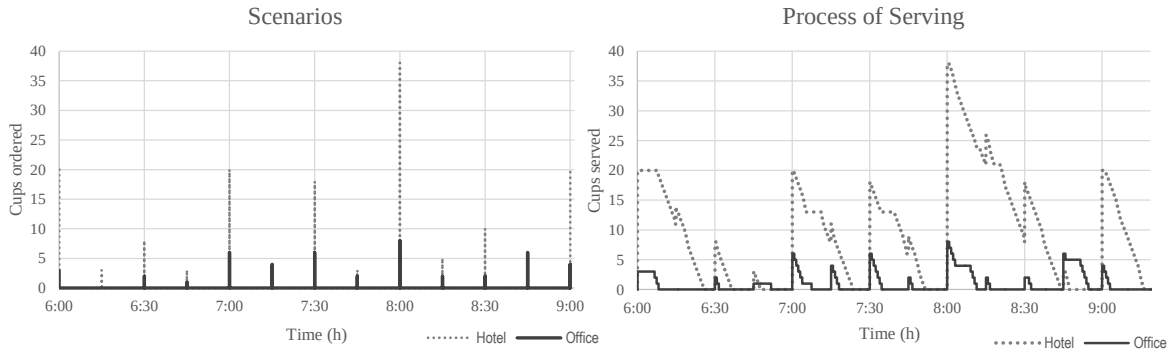

Figure 8. Scenarios; (a) Utilization (b) Process of serving

The functional analysis model stores every new cup order and calculates the actual number of cups being in waiting for the disposal. Thereby, the overall waiting time is calculated and acts as the objective function for the effectiveness of the service product. Comparing those two scenarios, a difference is noticeable not only in the peak times but also in the actual amount of cups ordered.

Therefore, an influence due to the necessary overall capacity, as well as the timing of the brewing and disposal cycles is expected. Due to the higher number of cups ordered in the hotel scenario, the main focus lays on a low machine operating time and therefore a low waiting time for the customer. This is investigated in a single objective optimization.

Although, the office scenario also requires a low operating time, the number of ordered cups, as well as the waiting time in between, is significantly lower (Figure 8 (b)). A single objective optimization for the lowest waiting time would lead to an oversized machine. Therefore, in the office scenario a multiobjective optimization is used, taking the contrary objective of water consumption of the machine into account, leading to a balance between the serving time and the tank sizes.

Figure 9 (a) shows the objective function space as scanned by the optimization algorithm for the hotel scenario. The number of evaluations is restricted to 300, resulting in an optimization time of 5 hours. Throughout the several compared optimization algorithms the Genetic Algorithm showed the best performance in this case. Already with this rather small number of evaluations the shape of the objective function space shows a definite trend, resulting in a minimum (circled data point), though no guarantee of a global optimum is given. For a better visualization, the three dimensional objective function space is shown with the buffer tank volume on the x-axis the brewing chamber volume on the y-axis and the overall operating time on the z-axis. These volume parameters are calculated retrospectively from the

https://doi.org/10.1017/dsd.2020.323 Published online by Cambridge University Press

variety of the geometric parameters. The optimum was found with a brewing chamber volume of 1.58 l and a buffer tank volume of 4.25 l, resulting in an operating time of 2.33 hours to serve all customers.
The multiobjective optimization of the office scenario leads to the diagram shown in Figure 9 (b) with several optimal solutions, resulting in a pareto-front (circled data points). The optimization algorithm used in this case is the Non-dominated Sorting Genetic Algorithm-II, again with a maximum number of 300 evaluations. It is apparent, that the pareto-front is not well developed and many evaluations are computed in higher objective function values. This can be explained due to the high number of varied parameters, leading to the conclusion that a greater number of evaluations is necessary to obtain a higher resolution pareto-front. Nonetheless, several optimal solutions are found, ranging from water consumption values of about 14 l to 12 l, while having an operating time from 0.75 hours to 1.05 hours. These values are obtained with tank volumes ranging from 1.1 l to 1.5 l for both tanks.

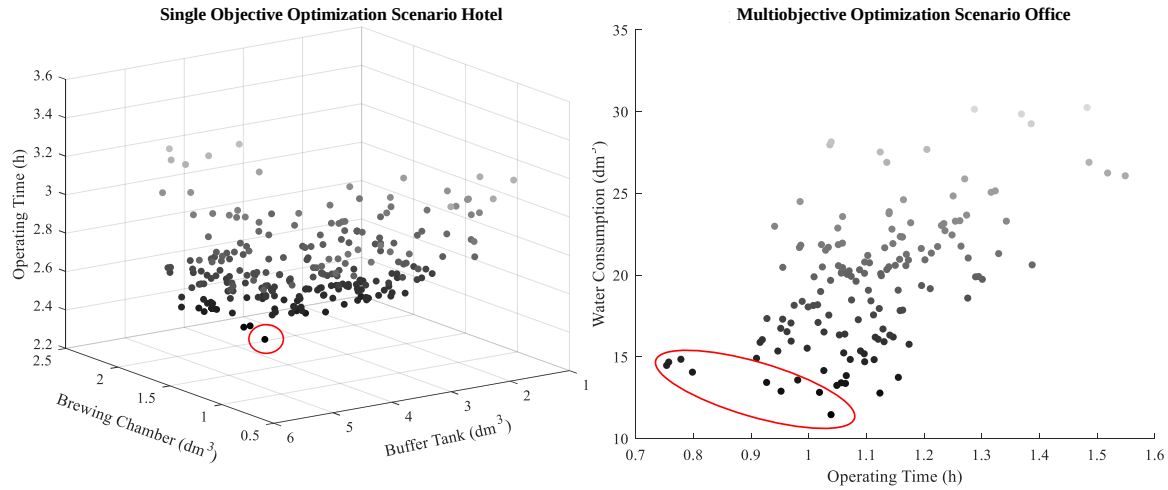

Figure 9. (a) single objective optimization of the scenario hotel (b) multiobjective optimization of the scenario office

The main focus of this optimization in this first study lays on the rather simple example of changing the tank sizes, as they bear a high sensivity for the depicted objective functions. However, the discrete changeable components, e.g. in form of the pumps showed good results, as they were exchanged to a higher power class while optimally filling the available design section.
As can be seen, this simple example of varying parameters in regard to the operating time and the water consumption already leads to a high difference in the parameter values and therefore in the overall machine design.

5. Conclusion and outlook

The presented research proposes an approach on the automated optimization of a product in user-specific scenarios. Using the generative design approach in combination with a functional analysis model, the use case of a coffee machine is simulated, while enhancing the possibility of modeling all necessary dependencies robustly. Additionally, the flexibility of changing parameters for continuously changeable components is given, as well as exchanging entire discretely changeable components inside specified design sections. The presented use case of the coffee machine shows promising results for the dimensioning and the exchange of certain components. The number of design parameters has a high influence on the necessary number of evaluations, as depicted in the multiobjective optimization of the office scenario. Here, the conclusion can be drawn that a higher number of iterations, and therefore higher amount of time, is necessary for the optimization cycle. Still, the described separation of domain and process knowledge, as well as the separation of geometrical and functional models, shows a highly flexible and robust overall model. Future work lays in the investigation of optimization hyperparameters for a more effective optimization, as well as the implementation of further objective functions like the cost. In addition, verification work beyond the example of the coffee machine has to be concluded.

https://doi.org/10.1017/dsd.2020.323 Published online by Cambridge University Press

Acknowledgment

This research was conducted in the scope of the research project SmartHybrid – Product Engineering (ID: 85003608) which is partly funded by the European Regional Development Fund (ERDF) and the State of Lower Saxony (Investitions- und Förderbank Niedersachsen NBank). We also like to thank the Institute of Structural Analysis from the Leibniz University Hannover for the provision of the optimization framework.

References

Bryant Arnold, C.R., Stone, R.B. and McAdams, D.A. (2008), "Memic: An interactive morphological matrix tool for automated concept generation", *IIE Annual Conference and Expo*, pp. 1196-1201.

Copeland, J. (2000), "The turing test", *Minds and Machines*, Vol. 10 No. 4, pp. 519-539. https://doi.org/10.1023/A:1011285919106

Du, K.-L. and Swamy, M.N.S. (2019), *Neural networks and statistical learning, Second edition*, Springer, London, United Kingdom. https://doi.org/10.1007/978-1-4471-7452-3

Gembarski, P.C., Bibani, M. and Lachmayer, R. (2016), "Design catalogues: Knowledge repositories for knowledge-based-engineering applications", *Proceedings of International Design Conference, DESIGN*, Vol. DS 84, pp. 2007-2016.

Hubka, V. (1976), *Theorie der Konstruktionsprozesse*, Springer Berlin Heidelberg, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-81035-0

Hvam, L., Mortensen, N.H. and Riis, J. (2008), *Product customization*, Springer, Berlin, London.

Koller, R. (1994), *Konstruktionslehre für den Maschinenbau*, Springer Berlin Heidelberg, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-08165-5

La Rocca, G. and van Tooren, M.J.L. (2010), "Knowledge-based engineering to support aircraft multidisciplinary design and optimization", *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, Vol. 224 No. 9, pp. 1041-1055. https://doi.org/10.1243/09544100JAERO592

Li, H. and Lachmayer, R. (2018), "Generative Design Approach for Modeling Creative Designs", *IOP Conference Series: Materials Science and Engineering*, Vol. 408, pp. 12035. https://doi.org/10.1088/1757-899X/408/1/012035

Meier, H., Roy, R. and Seliger, G. (2010), "Industrial Product-Service systems-IPS^2", *CIRP Annals - Manufacturing Technology*, Vol. 59 No. 2, pp. 607-627. https://doi.org/10.1016/j.cirp.2010.05.004

Mescheder, B. and Sallach, C. (2012), *Wettbewerbsvorteile durch Wissen: Knowledge Management, CRM und Change Management verbinden*, Springer, Berlin Heidelberg. https://doi.org/10.1007/978-3-642-27896-9

Milton, N.R. (2007), *Knowledge acquisition in practice, Decision engineering*, Springer, London.

Pahl, G. et al. (2007), *Engineering design: A systematic approach*, 3. ed, Springer, London. https://doi.org/10.1007/978-1-84628-319-2

Sabin, D. and Weigel, R. (1998), "Product configuration frameworks-a survey", *IEEE Intelligent Systems*, Vol. 13 No. 4, pp. 42-49. https://doi.org/10.1109/5254.708432

Sauthoff, B., Gembarski, P.C. and Lachmayer, R. (2016), "Maturity-model-based design of structural components", *Proceedings of International Design Conference, DESIGN*, Vol. DS 84, pp. 503-512.

Schreiber, D., Gembarski, P.C. and Lachmayer, R. (Eds.) (2018), *Developing a Constraint-Based Solution Space for Product Service Systems*.

Schreiber, G., Wielinga, B. and Breuker, J. (1993), *KADS: A principled approach to knowledge-based system development / edited by Guus Schreiber, Bob Wielinga, Joost Breuker*, Academic, London.

Thomas, O., Walter, P. and Loos, P. (2008), "Product-Service Systems: Konstruktion und Anwendung einer Entwicklungsmethodik", *Wirtschaftsinformatik*, Vol. 50 No. 3, pp. 208-219. https://doi.org/10.1365/s11576-008-0048-7

Tukker, A. (2004), "Eight types of product-service system: Eight ways to sustainability? Experiences from suspronet", *Business Strategy and the Environment*, Vol. 13 No. 4, pp. 246-260. https://doi.org/10.1002/bse.414

Wolniak, P., Sauthoff, B. and Lachmayer, R. (2018), "Scaling of Structural Components by Knowledge-Based-Engineering Methods", The Design Society, Glasgow, UK, May 21-24, 2018, pp. 1757-1768. https://doi.org/10.21278/idc.2018.0234

Wolniak, P. et al. (2019), "Scaling of Technical Systems Using an Object-Based Modelling Approach", *Proceedings of the Design Society: International Conference on Engineering Design*, Vol. 1 No. 1, pp. 1603-1612. https://doi.org/10.1017/dsi.2019.166

Zeigler, B.P., Praehofer, H. and Kim, T.G. (2000), *Theory of modeling and simulation: Integrating discrete event and continuous complex dynamic systems*, 2nd ed, Academic, San Diego, Calif., London.