INTERNATIONAL DESIGN CONFERENCE - DESIGN 2018
https://doi.org/10.21278/idc.2018.0409

EXECUTABLE COST-SENSITIVE PRODUCT DEVELOPMENT OF A SELF-BALANCING TWO-WHEEL SCOOTER WITH GRAPH-BASED DESIGN LANGUAGES

F. Wünsch, M. Ramsaier, T. Breckle, R. Stetter, M. Till and S. Rudolph

Abstract

This paper deals with the integration of cost considerations into the technological design process of a product. As an example serves a self-balancing two-wheel scooter. The design process of that scooter is expanded with aspects from the cost-domain so that the design can be enhanced with economic knowledge. Methods of modern software engineering are applied onto classical engineering. For this purpose so called graph-based design languages are used. Through the representation as an UML-model, the realization of interfaces to antecedent and subsequent systems can be eased.

Keywords: digital design, design automation, design knowledge, cost modelling, graph-based design language

1. Introduction

A large share of the activities in mechanical engineering is geometry-based and –centred. Additionally, a variety of software tools is used in the design process and the process integration between these software tools works only partially. A large part of the information which are required in the whole design process does not exist in the Computer-Aided-Design-(CAD)-models. Currently, a general trend in engineering towards individual production with a variety of individual product versions can be observed. To allow an efficient design of this multitude of individualized products, the automation of design process is necessary. Important is also the reusability of the rich body of knowledge in the design process. This representation of knowledge is often limited to geometrical information and is therefore underrepresented in existing models. As in the fourth industrial revolution automation and digital depiction of processes and thus the overcoming of interfaces gains importance, the competitiveness of companies depends exactly on these.

The challenge in engineering design is the integration of knowledge from different domains. Therefore, methods of software engineering like Design Patterns, abstraction and modelling languages may be applied to engineering design processes.

Product developers and designers often focus on technological aspects of their product; financial aspects are frequently not with considerable effort. Due to the massive influence of product costs in the early stages of the product lifecycle – especially in the planning and development phase – the product developers and designers have direct influence on the final product costs and should therefore know how their activities impact the costs of the product (Ehrlenspiel et al., 2014). As a last resort an evolutionary product would be possibly discarded for lack of efficiency after having carried out a cost analysis. Valuable work time is wasted because a rescheduling brings at least as much work as a redesign would require.

Previous research concerning graph-based design languages did not include deep considerations of cost calculation. However, graph-based languages dispose of the potential to interlink knowledge from different domains such as product geometry and product and context costs; this aspect is the main focus of this paper.
The described research is generally based on the Design Research Methodology (DRM) framework as presented by Blessing and Chakrabarti (2009). In this framework, four general stages of research can be distinguished. The presented research focuses on an investigation of a novel approach: graph-based design languages for cost representation in highly integrated product models. This can be characterized as a research project type five (Blessing and Chakrabarti, 2009). Here a product development support is created on the basis of a detailed generation of product development process understanding and is initially evaluated. Obviously, the research type five cannot really prove a research thesis, but it can indicate that the application of a product development support is logically sound and carries process improvement potential.
The next section presents the current state of the art of graph-based design languages, product cost calculation and representation of knowledge in libraries. The detailed description of the integration of the consideration of costs will be presented in Section 3. Section 4 describes the sample product, a self-balancing two-wheel scooter. The paper closes with discussion and reflection as well as a summary and outlook.

2. State of the art

2.1. Design methodology

The design process of a product is often long and iteratively. It represents the synthesis of the final design, which considers early stages specified definitions of the designed object. Requirements and constraints, which occur within the design process, have to be fulfilled (Lawson, 2006). In this case, constraints are defined as limiting conditions or restrictions revealing within the design process.
The presented approach within this paper uses the design methodology of Pahl and Beitz (Feldhusen and Grote, 2013). Considering that, the design process can be split up into these steps: Planning and task clarification, conceptual design, embodiment design, detail design and overall design process. In the time of the conceptual design phase, a principle solution is fleshed out. Therefor all essential problem statements are disassembled into subtasks (e.g. sub functions). Searching for suitable solutions or to use existing experience and knowledge is the goal. A combination of partial solutions leads to different solution alternatives. Afterwards, on the basis of defined evaluation criteria (e. g. technical or economic criteria) in order to find the most suitable solution the solutions alternatives are evaluated (Feldhusen and Grote, 2013). Complex design tasks call for iteration loops (Lindemann and Ponn, 2011; Feldhusen and Grote, 2013). An increasing complexity of the designs as well as a target improvement in efficiency postulates computer aided methods for automation within the design process (Feldhusen and Grote, 2013).

2.2. Graph-based design languages

In the state of the art design process a multiplicity of different domains do exist. This causes immense efforts and leads to an interface design and standard for exchanging data between these domains. A novel approach is to use a common language to describe the product lifecycle by means of the Unified Modeling Language (UML). By creating vocabularies (UML classes) and rules (UML activities) a design language can be generated for an arbitrary product. The result is a central data model in form of a design-graph (embodying the UML model) with one interface to each domain of the product lifecycle. Another advantage of using graph-based design languages is the inherent consistency of the generated graph as all domains gain their information from that one data model (single source of truth). Further forms of representation can be derived from the graph as it is shown in Figure 1: for example the geometry in form of a 3D model can be generated or a bill of materials (BOM) can be set up because all information that evolve during the runtime of the program is present in the graph. Also the product requirements are stored in the central model. Product requirements significantly define the manifestation of the product and have a massive impact on the geometry as well as the BOM and the product costs.

Deeper analysis of the representation of requirements in graph-based design languages conduct were carried out by Holder et al. (2017a). This integrated representation of many domains underlines again the advantages of the application of graph-based design languages. Classical CAD-tools presently do not provide such possibilities.

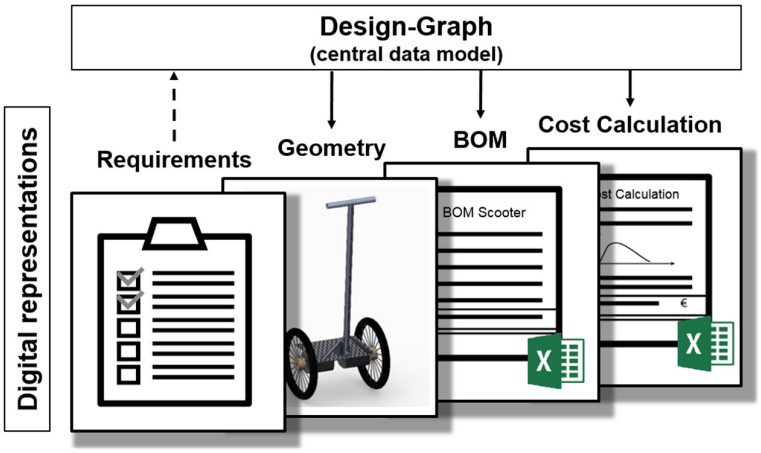

Figure 1. Central model with its forms of digital representations

It is important to note that the concept of graph-based design languages does not aim to replace any existing domain-specific software solutions but to integrate them. A central abstract data model can be transferred to a domain-specific view. Also the way back is possible, e.g. when the finite element simulation (as a domain specific view) has been solved and the results are fed back into the central data model. With the flexibility of this concept, proprietary software solutions and their model formats do not tie the user to their license system any more, as simply another interface can be written (e.g. for an open source solution) and no knowledge is lost. Currently all knowledge stored in proprietary file formats is lost, when the license is no longer available. Especially small and medium size businesses can highly profit from this approach as they cannot afford the high sophisticated analysis tools.
The idea of using graph-based design languages for engineering design comes from Rudolph, which he developed within his habilitation thesis (Rudolph, 2002). Synthesising on prior works of Lindenmayer, who uses simple rules to assemble complex structures (Lindenmayer, 1968) and developing a new algorithm, which find the right order of solving equations, Alber and Rudolph (2004) show the potential of rule-based engineering on the example of a power pole. This idea is further developed towards the development of a complete satellite system (Groß, 2013) and is currently explored within the research project ZAFH "Digital Product Life Cycle". Reichwein (2011) deals with the question, how UML can be a language for representing engineering objects, which leads to a shift away from XML to UML and is still today the modelling language of choice for graph-based design languages.

2.3. Product cost calculation

Product cost estimation and prediction at the early design phase is one of the main crucial aspects for a company to be successful due to shortening product life cycles and competitive markets. This allows a company to rapidly compare different solutions within the design process and make well-grounded decisions. An essential part of cost management is the product development accompanying calculation. To identify the impact of design decisions regarding the cost structure, a product cost calculation should be directly performed while designing (Ehrlenspiel et al., 2014). Within this early design stage, on the one hand the influence of the product design is great, but on the other hand the level of information of the future product is low (dilemma of development accompanying calculation). This leads to uncertain cost statements. In the interest of an early cost sensitive design process the inaccuracy must be accepted (Joos-Sachse, 2014). The process of predicting the cost of a work activity or an output by an interpretation of historical data or knowledge is known as cost estimation. Usually, this is done by cost modelling (Curran et al., 2004). The goal of a development accompanying cost calculation is to collect,

structure and process all relevant cost data and information in order to provide the results to designers, controllers and deciders. Methods and approaches for cost estimation have been presented in literature (e. g. in Duverlie and Castelain, 1999; Rush and Roy, 2000; Niazi et al., 2006). Those cost estimating or Design-to-Cost methods and approaches can be classified as intuitive methods, parametric techniques, variant-based models and generative cost estimating models (Nepal et al., 2008). The most used methods are based on knowledge, features, operations, weight, material, physical relationship and similarity laws (Weustink et al., 2000). According to the complexity of the mentioned methods and the required information basis a computer-aided support is helpful. In the literature and industrial application, there exist many IT solutions, but mostly developed for a specific use case (Mörtl, 2013). If the level of detail within the product development process increases, it results in more accurate product cost calculations.
As the design process is iterative the cost estimation (or later in the process the cost calculation) needs often to be changed, updated or post-processed. If there is an uncoupled IT solution, the process is error-prone and connected with manual work. This reduces the acceptance as it is a time-consuming process and the data consistency and traceability cannot be ensured.

2.4. Representation of knowledge in libraries

To integrate the modelling of product costs effectively into the product design, it is reasonable to store the cost-structures in an abstract manner in a library as it is known from programming languages. In the IEEE-standard a software library is defined as "a controlled collection of software and related documentation designed to aid in software development" (IEEE, 1990). It is comparable to a kind of modular system. So it is possible for every modeller (and designer) to reuse that knowledge from the library whenever it is necessary to model product costs for example. The UML as a modelling language provides the optimal environment for that. On the one hand the reusability is supported through the object-oriented approach and on the other hand the UML-model can depict a large share of the knowledge which is used in the product development as a central and holistic model. Through this knowledge representation in the model third parties are able to understand the thoughts of the designer and can retrace the product design. In addition to this, it is possible to reuse the knowledge which is included in that specific product design. Feldhusen et al. (2013) mention so called product data management systems to enrich a product with further information and knowledge. Chapman and Pinfold (1999) describe a similar approach of knowledge reuse by applying knowledge-based engineering (KBE).
A characteristic of knowledge modelling in the central model is that the knowledge is not depicted encapsulated by functions, but it is modelled in a cross-functional manner for the product. This fact both demands and supports the collaboration of specialists in different functional areas and is harmonious with the process-oriented point of view.
To support the representation of knowledge, it is reasonable to create libraries for the specific functional knowledge. These libraries provide the necessary components for the knowledge representation in the model and are universally usable. So it is possible to apply this library on every project that is created by graph-based design languages. For instance, a geometry library for graph-based design languages is applied in various works like Ramsaier et al. (2017b), Wünsch (2016), Groß (2013).
Each library itself is a graph-based design language. Therefore, it contains its own class-diagram. In this diagram the necessary classes are created with their inheritance relationships and the associations with each other (as usual in graph-based design languages). Thereby the blueprint is created how the costs can be modelled later in the project. For the product designer it is necessary to know the content of the class diagram of the library to be able to model in the right manner.
Any such library can be integrated into a design project using graph-based design languages. With this one has the possibility to use the classes of the library and create instances of these in the own model.

3. Integration of consideration of costs

The specific objective of the integration of costs is to extend engineering design beyond technological or physical aspects. The research aims at the integration of economic considerations which lead to the specific product. As mentioned at the beginning, the great advantage by using graph-based design

languages is that changes of a model are possible at any time, because it can be computed after changes in the model in shortest time. Thereby the engineer does not have to renew all designs of all domains that are affected by the changes of the model since these changes not only influence the emerging geometry of the product but also the costs for example. There is a big potential to evaluate a model also by means of the evolving product costs to provide a decision support.
The consideration of the costs already in the model of the product design makes sense because it is possible to achieve cost predictions for the entire product. What kind of costs arises and in which amount do they accumulate?
Reflections of economical design (Ehrlenspiel et al., 2014) or of performing the cost accounting in-design (Joos-Sachse, 2014) were already initiated. It is a well-known fact that especially during the product design phase there exist many possibilities to influence the product costs.
On the downside it is a problem that at this early stage little information about the evolving product is present. The result of a cost calculation is the more precise the more details of the design of the product are known. Especially the choice of the concept, the material and the manufacturing processes play a big role because these are the influencing criteria for the resulting costs. Another problem is that with the increasing level of detail the calculation results are not improving analogous to the higher information level and thus there is a change of methods for the cost estimation procedure that often leads to difficulties as for example misinterpretation (Joos-Sachse, 2014).
It is a main thesis of the presented research that these problems can be addressed by using graph-based design languages for the design of a product. Since the costs are included in the model in form of UML-instances they are on the one hand transparent at any time and on the other hand always fit to the level of detail of the model. The cost information develops analogous to the proper technical product design. The additional effort for the consideration of costs during the development process do not cease even though, but after having modelled the costs the cost calculation is adjusted automated at every change of the model without additional effort. So no separate post-calculation is necessary if the cost calculation is kept consistently in the model.
The majority of the costs of production factors is determined by the manufacturing processes applied because with the choice of a special manufacturing process it is obvious which operating materials and which workforces are necessary. Therefore, manufacturing processes have to be part of the cost model, which leads to an enlargement of information and the representation of engineering knowledge.

3.1. Cost-library

As mentioned before a cost library was developed in form of a graph-based design language for modelling costs during the product design process to support the cost calculation and generate cost forecasts for the product. The development of the design language for cost modelling was oriented on the production factors oriented approach of cost accounting (compare Section 3). On this basis it is established both which factors of the product manufacturing are used and what they cost. A possible approach of the production factors oriented treatment of cost categories according to (Walter and Wünsche, 2013) is illustrated in Figure 2.

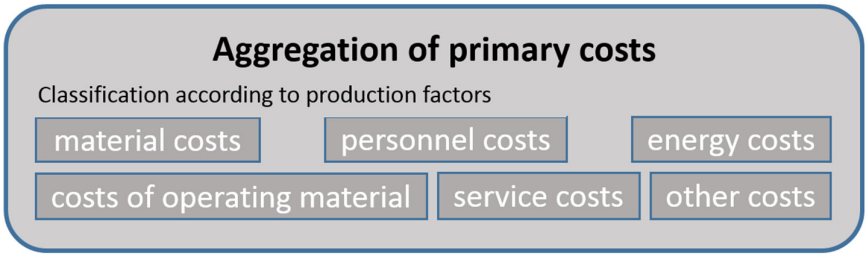

Figure 2. Production factors oriented treatment of cost categories

The dispositive production factors management, planning, organisation and controlling according to Gutenberg (1951), are not considered yet, but will be the focus of further research.

The chosen approach results in material costs, personnel costs, costs of operating material, energy costs, service costs and other costs. According to the domain other differentiation would be possible (Walter and Wünsche, 2013). But since the cost library should be applicable universally and not for a distinct application the consideration of the costs should take place on this abstract level.
In consideration of the costs of production factors and manufacturing processes, the UML class diagram that is illustrated in 0 was developed. This diagram along with the cost-plugin in form of Java code serves as cost library for the modelling of costs within graph-based design languages. The cost plugin analyses the modelled instances and creates a cost calculation out of them.
The class *CostModel* (red in Figure 3) serves in the program as an entry into the model for generating the cost calculation. This class has access to all cost components (grey in Figure 3) through the associations (blue arrows). For the actual modelling of costs this class can be neglected. The classes *CostComponent*, *CostComponentUnit* and *CostComponentDuration* (orange in Figure 3) represent super-classes from which all cost components can inherit the necessary attributes. Since they are abstract classes no instances of them can be modelled. The cost component classes are derived from the costs for production factors which were mentioned before. It can be distinguished between the following classes: *PurchasedPart* represents all parts that are not of the own manufacture, *SemifinishedProduct* stands for a semi-finished product parts of a distinct material, for example a stainless steel tube. To define the (raw-) material the respective material has to be modelled by means of the material library which contains all kind of materials and its parameters that are important for a product design. This library was developed parallel to the cost library as it became clear that product costs are strongly influenced by the used materials. For example a component made of carbon can be more expensive than one made of aluminium. To serve the importance of materials with regard to the product costs, they shall be represented explicitly in the model. Furthermore the modelling of the materials represents engineering knowledge of the product design. So a separate material library was created for the support of the cost modelling of an example product described below. This library contains possible materials in form of UML classes which can be provided with all attributes that are necessary for the modelling. These classes can be instantiated and used in combination with classes of the cost library.

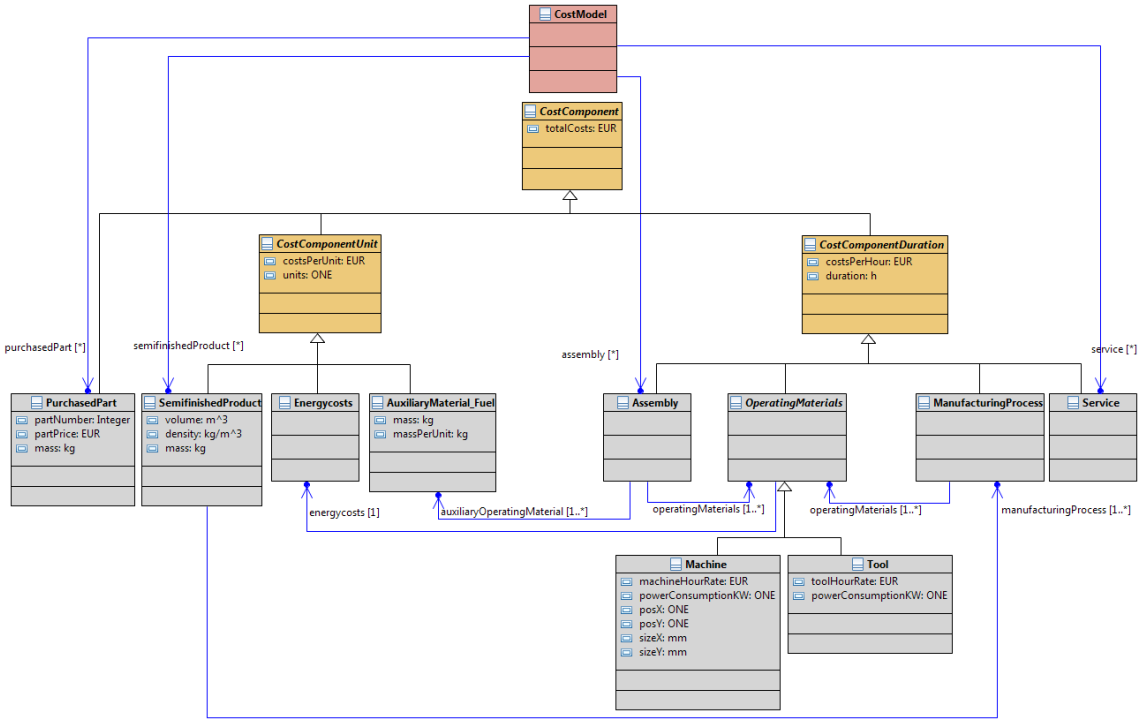

Figure 3. Class diagram of the cost library

The class *Energycosts* represents costs for any kind of (process) energy like for example electrical energy or gas energy which arise in the usage of operating materials. Because of the arising importance of the energy balances of products, the modelling of energy costs can support the schedule of the energy balance which represents its own domain and has to be seen separately from the costs. *AuxiliaryMaterial_Fuel* represents any kind of both auxiliary materials and fuels. These are combined in one class because in practice the border between them is often unclear. The class *Assembly* represents human work force within an assembly operation. Since the process in the lifecycle shall be modelled in a transparent manner this special class was chosen instead of personnel costs for example. For an assembly, one or more auxiliary materials, fuels or operating materials may be necessary. This fact is represented by the associations going out of this class. *OperatingMaterials* is an abstract class from which all possible operating materials can inherit. In the current model the operating materials *Machine* and *Tool* are implemented, but it is possible to extend the cost library by additional operating materials like properties, buildings, means of transport or office equipment (Wöhe and Döring, 2002). An operating material can be necessary for example for a manufacturing process or for an assembly. The class *ManufacturingProcess* similarly to *Assembly* represents human work force during that process. Since the manufacturing process determines a lot of cost factors and is of importance for the subsequent production planning and the planning of the digital factory, it is very important to implement it in the cost library. Finally the class *Service* represents all kind of services that are necessary in the process to be modelled. Nowadays services have become very important in a product lifecycle and more activities are outsourced from the manufacturer to service providers.

3.2. Cost modelling

The cost library can be integrated into the design language of the example product described below in Section 4.

Figure 4 shows the UML activity diagram in which the different domains of the product are modelled: the requirements, the topology of the product with the resulting geometry and the product costs with the aggregation of the costs at the end of the program sequence which finally creates the cost calculation of the product. So the knowledge of all these domains is pooled in this design-graph which is shown in the bottom right of Figure 5.

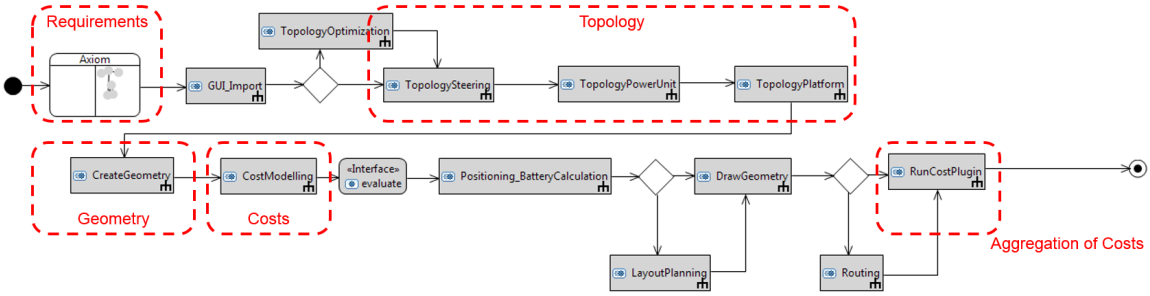

Figure 4. UML activity diagram: Requirements, topology, geometry and costs

The Sub-Program *CostModeling* contains all UML transformation rules in which the model is enriched by components of the cost library. As an example of the cost modelling serves the bar - a part of the example product. The rule for the cost modelling of the bar is shown in the upper right of Figure 5: an existing instance of the class *bar* is equipped with instances of the cost library within this rule. With the execution of that rule the graph, which represents the holistic model with all its domains and links to each other, grows simultaneously and the cost instances are added to the bar as new nodes. The difference of the graph of the model before and after the cost modelling (upper left and bottom right in Figure 5) can be seen very clearly. That underlines the importance of the additional cost-information that a component can be provided with.

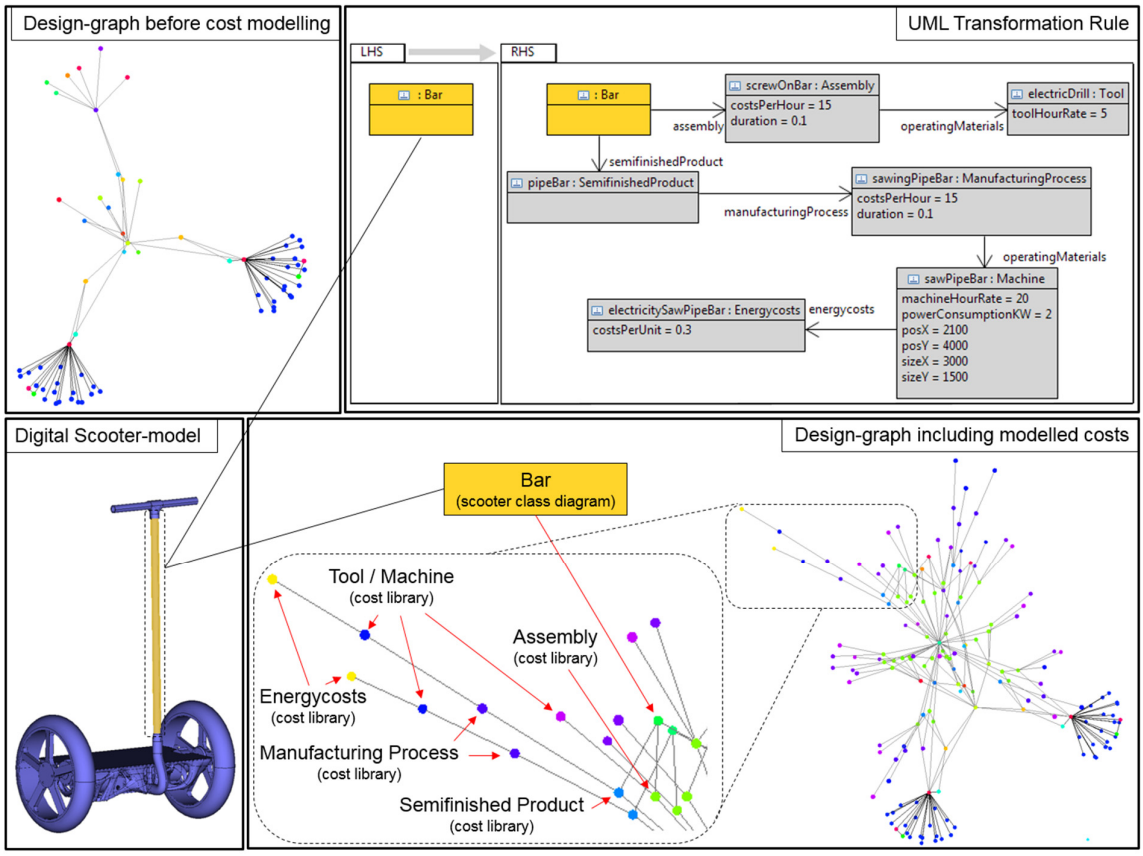

Figure 5. Cost modelling: UML instances and model representation as a design-graph

During the execution of the program the cost plugin of the library creates an excel worksheet within which all modelled cost components are listed clearly, broken down by cost component.
This leads to the bill of materials as mentioned at the beginning. Also the BOM can be generated by the described cost plugin. A multi-level bill is generated and is enriched with information about costs and weight of the components. The BOM is very important for the stochastic material requirements planning and the cost calculation of a product. It has not to be created manually any more: All instances of the central model which represent a component of a product are automatically inserted into the BOM. Since all further information from other domains is deposited at the relevant instance, the BOM can be easily enriched with more information. A change of the model leads to a change of the BOM. So it is ensured that the BOM is always filled with consistent data - a media disruption can be prevented.
The more detailed the costs are modelled, the more precise can be the cost forecast for the product. Since the focus of the research project is on the design process, not all kinds of costs that evolve during the whole product lifecycle are mentioned. Production costs are in the focus because the design process is closely linked with the production process. However, with this method of cost modelling it is possible to represent any cost type of the product lifecycle. The cost library can easily be extended.
By modelling costs within the design process of a product, the costs are allocated to individual cost centres. This contributes to the aim to connect the costs directly with the performances so that an optimum planning, management and controlling can be achieved to secure the corporate success.
Another advantage of the cost modelling during the design process is the fact that already at this time in the product lifecycle it is possible to monitor the probable accumulated product costs by means of the model. This for example can lead to the realisation that the predicted costs are not within the prognosticated scope. In that case an early reaction is possible, for example the adaption of the design

to fit the budget or an early information about the prognosis, which can be included in further decisions. This in turn can lead to a saving of money and effort in later phases of the product lifecycle.

4. Example product: Self-balancing two-wheel scooter

As an application example a self-balancing two-wheel scooter has been chosen within this work. The scooter also serves as a use-case of the research project "Digital product life cycle" (ZAFH DiP). A basic model of this scooter was developed in a recent master thesis (Wünsch, 2016). Based on this, a model was created which combines the geometry with the costs and the materials of the described libraries. The result of the cost modelling can be seen in Figure 7. According to the used cost components and materials of the libraries the costs and weights can be listed in an excel file. It is also possible to present individual costs and weights for all components of the products. The calculation of product variants is no problem: for example 100 different configurations of a scooter can be generated by changing materials of the components which also leads to a change of manufacturing processes and different working times. The computation of these variants is not such a time consumable process so that many variants can be generated and compared.
That in turn offers the possibility of optimization loops to get the best product valued by quality, cost or design aspects.
Figure 6 shows four different scooters which were all generated by means of the same graph-based design language which was developed within the research project. They differ in the use of different materials, batteries, manufacturing processes, etc.

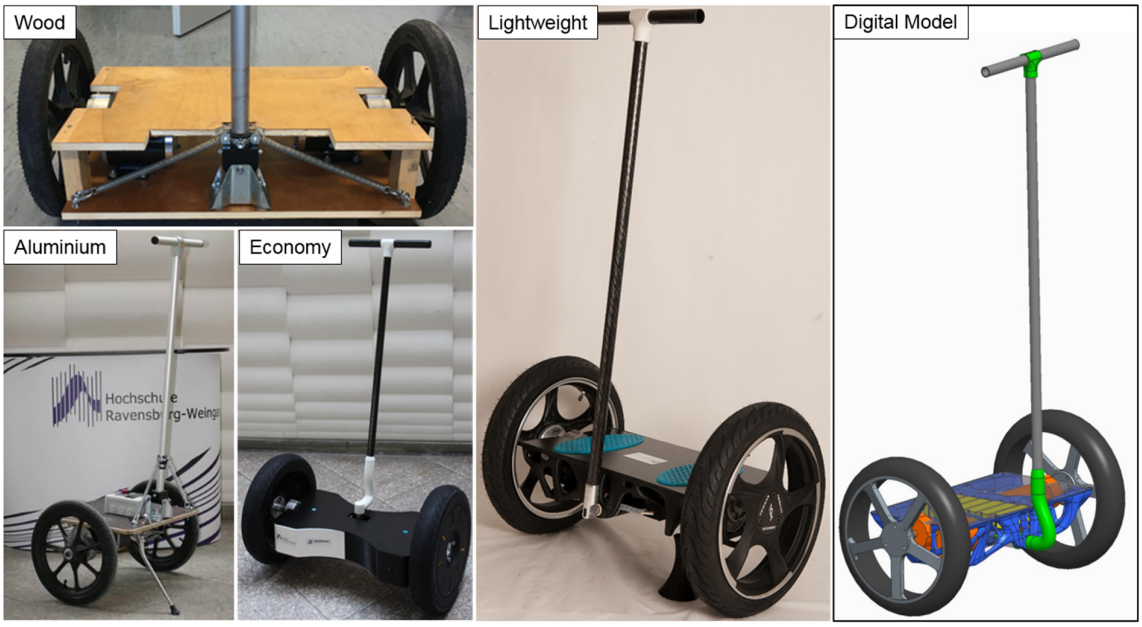

Figure 6. Manufactured variants of the scooter

For the prove of the methodology of the cost modelling, five variants of the scooter were generated: a model made of wood with cheaper engines and batteries, one made of aluminium parts, one made of mainly carbon parts, one made of mainly self-printed 3D-parts and one with a professionally printed 3D-frame based on an automated topology optimized structure (Wünsch, 2016; Burkhart, 2017; Schuster and Pahn, 2017). Figure 7 shows the composition of the costs of these five variants by means of the cost modelling.

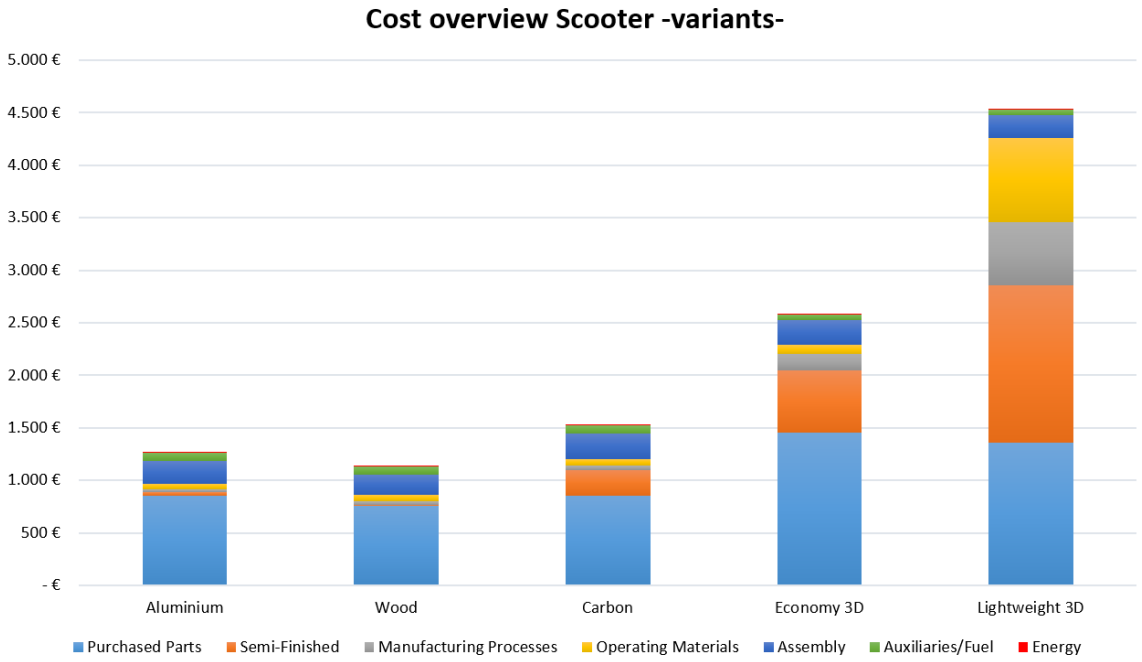

Figure 7. Cost overview of five variants of the scooter

5. Discussion and reflection

Before using the design language for costing, the cost modelling was created manually in Excel. This required a manual adjustment of the cost calculation whenever the design was changed or another variant was designed. This was error-prone and time consuming due to manual data synchronization and thus not always consistent. By the idea of the central data model within the graph-based design languages not only each product configuration has a consistent design but the cost model is always synchronized with the current product design. This approach leads to greater cost sensitivity, as the costs are always up-to-date and directly visible. By using graph-based design languages with integrated cost modelling it is therefore possible to represent a self-balanced two-wheel scooter including several domains. This is especially remarkable, as the resulting system allows a dynamic development of new product configurations which are not only parametrical but also topological different. With this system, individualized scooter variants can very quickly be developed taking into consideration individual size and weight as well as individual requirements concerning operation range; maximum speed; maximum acceleration. For these configurations detailed cost information is instantly available. This also allows identifying less desirable configurations. Especially in connection with additive manufacturing methods new possibilities for individualized products are evolving. In this example graph-based design languages allow multi-domain modelling to an extend which cannot be achieved with conventional modelling methods - detailed cost calculation models are directly connected with elaborate simulations concerning durability and stiffness as well as electrical consumption. Currently the same methodology is also applied to complex product components in automotive and transportation industry (Holder et al., 2017a; Holder et al., 2017b; Ramsaier et al., 2017b) and multicopters (Ramsaier et al., 2017a). Integrating the design of the associated assembly system for this self-balanced two-wheel scooter can improve the informational value and the accuracy of the cost estimation and calculation, as applied to automotive production systems (Breckle et al., 2017). The reflection of the processes with the involved designers leads to the insight that a further development of the applied systems is desirable, but also that general objectives can be achieved.

6. Summary and outlook

The main thesis of the presented research that the problems arising with the complexity of cost consideration and the necessity of early cost consideration can be addressed by using graph-based design

languages for the design of a product is supported by the presented results. Further research work is needed (and planned) for an expansion of this indication. Additionally, further research will concern the integration of a commercial cost modelling program (Siemens Teamcenter Product Cost Management) and an expansion of the integrated cost model in the direction of further important aspects of the product life cycle such as ecological evaluation and recycling.
Also interfaces to databases or ERP-systems of organizations will have to be realised so that for example material prices are always up-to-date in the central model without the need of manual adaption.
Regarding the refining of manufacturing costs a library for assembly methods with an integrated calculation of unit times (predetermined motion time systems) requires research.

Acknowledgment

The project „digital product life-cycle (ZaFH)“ (information under: https://dip.reutlingen-university.de/) is supported by a grant from the European Regional Development Fund and the Ministry of Science, Research and the Arts of Baden-Württemberg, Germany (information under: www.rwb-efre.baden-wuerttemberg.de).

References

Alber, R. and Rudolph, S. (2004), “On a Grammar-Based Design Language That Supports Automated Design Generation and Creativity”, In: Borg, J.C., Farrugia, P.J. and Camilleri, K.P. (Eds.), *Knowledge Intensive Design Technology*, Springer, Boston, pp. 19–35. https://doi.org/10.1007/978-0-387-35708-9_2

Blessing, L.T.M. and Chakrabarti, A. (2009), *DRM, a Design Research Methodology*, Springer, London, UK. https://doi.org/10.1007/978-1-84882-587-1

Breckle, T., Kiefer, J., Rudolph, S. and Manns, M. (2017), “Engineering of assembly systems using graph-based design languages”, *Proceedings of the ICED’17 / 21st International Conference on Engineering Design, Vol. 4: Design Methods and Tools*, The Design Society, Glasgow, pp. 519–528.

Burkhart, J. (2017), Digitaler Entwurf und Bau eines Segways, Master thesis, Hochschule Ravensburg-Weingarten, Weingarten.

Chapman, C.B. and Pinfold, M. (1999), “Design engineering—a need to rethink the solution using knowledge based engineering”, *Knowledge-Based Systems*, Vol. 12 No. 5-6, pp. 257–267. https://doi.org/10.1016/S0950-7051(99)00013-1

Curran, R., Raghunathan, S. and Price, M. (2004), “Review of aerospace engineering cost modelling. The genetic causal approach”, *Progress in Aerospace Sciences*, Vol. 40 No. 8, pp. 487–534. https://doi.org/10.1016/j.paerosci.2004.10.001

Duverlie, P. and Castelain, J.M. (1999), “Cost Estimation During Design Step. Parametric Method versus Case Based Reasoning Method”, *The International Journal of Advanced Manufacturing Technology*, Vol. 15 No. 12, pp. 895–906. https://doi.org/10.1007/s001700050147

Ehrlenspiel, K., Kiewert, A., Lindemann, U. and Mörtl, M. (2014), *Kostengünstig Entwickeln und Konstruieren: Kostenmanagement bei der integrierten Produktentwicklung,* 7th ed., Springer Vieweg, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-41959-1

Feldhusen, J. and Grote, K.-H. (2013), *Pahl/Beitz Konstruktionslehre: Methoden und Anwendung erfolgreicher Produktentwicklung,* 8th ed., Springer Vieweg, Berlin. https://doi.org/10.1007/978-3-642-29569-0

Feldhusen, J., Grote, K.-H. and Thon, J. (2013), “Grundsätzliche Überlegungen zur Rationalisierung”, In: Feldhusen, J. and Grote, K.-H. (Eds.), *Pahl/Beitz Konstruktionslehre*, Springer Vieweg, Berlin, Heidelberg, pp. 773–815. https://doi.org/10.1007/978-3-642-29569-0_16

Groß, J. (2013), “Aufbau und Einsatz von Entwurfssprachen zur Auslegung von Satelliten”, PhD thesis, Institut für Statik und Dynamik der Luft- und Raumfahrtkonstruktionen, Universität Stuttgart, Stuttgart.

Gutenberg, E. (1951), *Grundlagen der Betriebswirtschaftslehre: Band 1: Die Produktion*, Springer-Verlag Berlin Heidelberg, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-21965-2

Holder, K., Zech, A., Ramsaier, M., Stetter, R., Niedermeier, H.-P. et al. (2017a), “Model-Based Requirements Management in Gear Systems Design Based On Graph-Based Design Languages”, *Applied Sciences*, Vol. 7 No. 11, pp. 1112. https://doi.org/10.3390/app7111112

Holder, K., Zech, A., Stetter, R. and Till, M. (2017b), “Ansatz zur rechnergestützten Synthese und Analyse von Entwurfsvarianten für Formula Student Getriebe mittels graphenbasierter Entwurfssprachen”, *ASIM-Treffen STS/GMMS 2017: Workshop der ASIM/GI Fachgruppen STS und GMMS*, *ASIM Mitteilung*, ARGESIM Verlag, Wien, pp. 268–273.

IEEE (1990), IEEE standard glossary of software engineering terminology IEEE Std 792-1983, Inst. of Electrical and Electronics Engineers, New York, NY.

Joos-Sachse, T. (2014), *Controlling, Kostenrechnung und Kostenmanagement: Grundlagen - Anwendungen - Instrumente,* 5th ed., Springer Gabler, Wiesbaden. https://doi.org/10.1007/978-3-658-01344-8
Lawson, B. (2006), *How designers think: The design process demystified,* 4th ed., Routledge, London. https://doi.org/10.4324/9780080454979
Lindemann, U. and Ponn, J. (2011), *Konzeptentwicklung und Gestaltung technischer Produkte: Systematisch von Anforderungen zu Konzepten und Gestaltlösungen,* 2nd ed., Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-20580-4
Lindenmayer, A. (1968), "Mathematical models for cellular interactions in development. Part 1 and 2", *Journal of Theoretical Biology*, Vol. 18 No. 3, pp. 280–315. https://doi.org/10.1016/0022-5193(68)90079-9
Mörtl, M. (2013), Kurzkalkulation – Literaturrecherche/-studie, FVA-Forschungsvorhaben Nr. 632 I, Heft 1043, Forschungsvereinigung Antriebstechnik, Frankfurt.
Nepal, B., Monplaisir, L., Singh, N. and Yaprak, A. (2008), "Product Modularization Considerıng Cost and Manufacturability of Modules", *International Journal of Industrial Engineering: Theory, Applications and Practice,* Vol. 15 No. 2, pp. 132–142.
Niazi, A., Dai, J.S., Balabani, S. and Seneviratne, L. (2006), "Product Cost Estimation. Technique Classification and Methodology Review", *Journal of Manufacturing Science and Engineering*, Vol. 128 No. 2, pp. 563-575. https://doi.org/10.1115/1.2137750
Ramsaier, M., Holder, K., Zech, A., Stetter, R., Rudolph, S. and Till, M. (2017a), "Digital representation of product functions in multicopter design", *Proceedings of the ICED'17 / 21st International Conference on Engineering Design, Vol. 4: Design Methods and Tools*, The Design Society, Glasgow, pp. 369–378.
Ramsaier, M., Stetter, R. and Till, M. (2017b), "Modellierung und Simulation eines Formula Student Rahmens mittels graphenbasierter Entwurfssprachen", *ASIM-Treffen STS/GMMS 2017: Workshop der ASIM/GI Fachgruppen STS und GMMS*, *ASIM Mitteilung*, ARGESIM Verlag, Wien, pp. 274–279.
Reichwein, A. (2011), Application-specific UML profiles for multidisciplinary product data integration, PhD thesis, Faculty of Aerospace Engineering and Geodesy, Universität Stuttgart, Stuttgart.
Rudolph, S. (2002), *Übertragung von Ähnlichkeitsbegriffen*, Habilitationsschrift, Universität Stuttgart, Stuttgart.
Rush, C. and Roy, R. (2000), "Analysis of cost estimating processes used within a concurrent engineering environment throughout a product life cycle", In: Ghodous, P. and Vandorpe, D. (Eds.), *Advances in concurrent engineering: CE2000*, Technomic, Lancaster, pp. 58–67.
Schuster, J. and Pahn, F. (2017), "Entwicklung und Bau zweier konzeptionell unterschiedlicher Segways", bachelor thesis, Hochschule Ravensburg-Weingarten, Weingarten.
Walter, W.G. and Wünsche, I. (2013), *Einführung in die moderne Kostenrechnung: Grundlagen - Methoden - Neue Ansätze Mit Aufgaben und Lösungen,* 4th ed., Springer, Wiesbaden. https://doi.org/10.1007/978-3-8349-4075-9
Weustink, I.F., ten Brinke, E., Streppel, A.H. and Kals, H.J.J. (2000), "A generic framework for cost estimation and cost control in product design", *Journal of Materials Processing Technology*, Vol. 103 No. 1, pp. 141–148. https://doi.org/10.1016/S0924-0136(00)00405-2
Wöhe, G. and Döring, U. (2002), *Einführung in die allgemeine Betriebswirtschaftslehre, Vahlens Handbücher der Wirtschafts- und Sozialwissenschaften,* 21st ed., Vahlen, München.
Wünsch, F. (2016), "Multidisziplinäre digitale Repräsentation des Produktlebenszyklus am Beispiel eines Segways", Master thesis, Hochschule Ravensburg-Weingarten, Weingarten.

Fabian Wünsch, MSc.
Hochschule Ravensburg-Weingarten, Institut für angewandte Forschung
Doggenriedstraße, 88250 Weingarten, Germany
Email: fabian.wuensch@hs-weingarten.de