NORDDESIGN CONFERENCE
REYKJAVIK, ICELAND, 12-14 AUGUST 2024

Methodology for Support of Battery System Development with Format-Flexible Pouch-Cells using Model-Based-Systems-Engineering

Yunying Zeng, Philip Müller-Welt, Katharina Bause

Karlsruhe Institute of Technology (KIT), IPEK – Institute of Product Engineering,
Kaiserstraße 10, 76131 Karlsruhe, Germany

Abstract: This paper explores the evolving complexity of multidisciplinary development processes in battery systems. Designing battery systems with format-flexible cells challenges traditional approaches by emphasizing iterative optimization and data exchange between design aspects. To cope with this challenge, we integrate Model-Based Systems Engineering to develop a descriptive model outlining the overall system development process, including its sub-steps and iterations. Additionally, we provide an illustrative application of the methodology on one design step as part of the overall design tool.

Keywords: Systems Engineering (SE), Model-Based-Systems-Engineering (MBSE), Product Development, Battery Systems, Design Support System

1 Introduction

1.1 Development of Battery Systems with Format-Flexible Pouch-Cells

Today's development processes are becoming increasingly complex due to the multidisciplinary nature of products. One example is the development of battery systems. This includes aspects of mechanics, thermics, electrics and chemistry (Bouvy et al., 2012) (Albers et al., 2014). There are also various approaches to the design, optimization and validation of battery systems. In a common battery system development process, suitable battery cells are first selected for the requirements of the application based on their geometry as well as electrical and thermal properties (Lensch-Franzen et al., 2020) (Rajasekhar and Gorre, 2015). The number of battery cells required can then be determined. The number of cells connected in series determines the system voltage achieved and the number of cells connected in parallel determines the system capacity achieved. At the same time, the packaging must also be considered. Often there is only a certain installation space that limits the arrangement of the cells and modules. Finally, the design of the cooling system and safety aspects can be considered. (Schmalz et al., 2020)

Format flexible cells could overrule these former boundaries and introduces a novel approach in the development of battery systems. This provides for a flexible design of the cell shape and production. This means that, in contrast to standard formats, individual cells can be designed variably in terms of length, width and thickness, for example. This approach offers new possibilities in the design of the entire battery system, such as improved utilization of the available installation space, which can mean an increase in the capacity achieved. However, it also presents new challenges in the design of the battery system. On the one hand, the packaging becomes more complex due to the significantly increased solution space as well as the multiple interdependencies between the different domains. At the same time, it must be possible to ensure a uniform electrical and thermal load on the individual cells to be able to utilize the full performance or capacity of the battery system on the other hand and to avoid differential ageing of the cells.

The approach described is being pursued by the authors as part of the project "*AgiloBat – Entwicklung eines agilen Produktionssystems für die format-, material- und stückzahlflexible Pouch-Zellen Produktion*" ("Development of an agile production system for format-, material- and quantity-flexible pouch cell production"), funded by the Baden-Württemberg Ministry of Science, Research and the Arts, in which both new approaches to the development of such battery systems and a production system to enable flexibility are being developed in the sense of Product-Production-CoDesign (PPCD) (Albers et al., 2022), (Ruhland et al., 2022). The overall optimization of such a system requires various iteration loops and case distinctions as well as the transfer of data between individual design aspects. The methodology pursued for this is described in more detail in (Gandert et al. 2024). This methodology involves some challenges during the development process. Support in the development process of such a complex system can be provided, for example, through the use of model-based systems engineering (MBSE).

1.2 Model-Based-Systems-Engineering

According to (Stachowiak, 1973), a model is a representation of natural or artificial originals. Among several common objectives, models serve as representations to facilitate understanding or analysis, to define or specify, to communicate or visualize, and to verify or validate systems (Walden et al., 2015). Different model types are involved in the development process, such as descriptive or predictive models, which are selected based on the specific purpose they are intended to fulfill.

Model-Based-Systems-Engineering is an approach to system design and development in which the creation and use of interconnected system model, a descriptive model captured in SysML, is an integral part of the development process (Walden et al., 2015). In MBSE, a system is represented by graphical and textual models that capture various aspects such as requirements, function, structure, behavior and interactions. The basic idea of MBSE is to use models to understand, analyze and document complex systems in a systematic and structured way. These models can be used to simulate and validate system behavior, analyze system architecture and ensure that requirements are properly captured and tracked throughout the development process (Weilkiens, 2006). It helps to manage complexity, improve communication between stakeholders and facilitate collaboration throughout the lifecycle of a system.

MBSE also offers the option of specifying system behavior, for instance mapping processes and depicting sequences of events over time. For this purpose, common MBSE tools offer the option of creating activity diagrams, which excel in expressing the flow of objects (matter, energy, or data) through a behavior and focus on representing how these objects are accessed and modified during system operation (Delligatti, 2014). This makes it possible, for example, to map an entire design process with its sub-steps and iterations. It is also possible to define interfaces between individual design aspects through which data or parameters can be exchanged. This can ultimately support development processes that require collaborative cooperation between different domains. This is ensured by a holistic insight into the interactions and interrelationships of the individual design aspects.

To cope with the complexity of developing and optimizing a battery system with format-flexible cells, we incorporate aspects of MBSE in our approach to develop a model that describes the design and optimization process. For this purpose, the software Cameo Systems Modeler by Dassault Systems is used to create a design supporting model implementing the methodology. The design supporting model represents the different steps and functions executed by the actual design tool, which is implemented in MATLAB. Based on (Friedenthal et al., 2012), the design supporting model to be developed serves as a descriptive model, providing a common and discipline-independent understanding, whereas the MATLAB tool functions as a predictive model, used to explain and simulate the behavior of the battery system. The followed methodology and derived requirements for the design supporting model are described in the following chapter. Afterwards an illustrative application of the methodology on the design step Installation Space Optimization as part of the overall design tool is shown.

2 Methodology for Supporting the Application of Design Tool

The overall design and optimization process of battery systems with format-flexible pouch cells is illustrated in **Figure 1 (a)** and described in more detail in (Gandert et al. 2024). To support the development and application of this design process, the methodology of creating the design supporting model is elaborated and is depicted schematically in **Figure 1 (b)**.

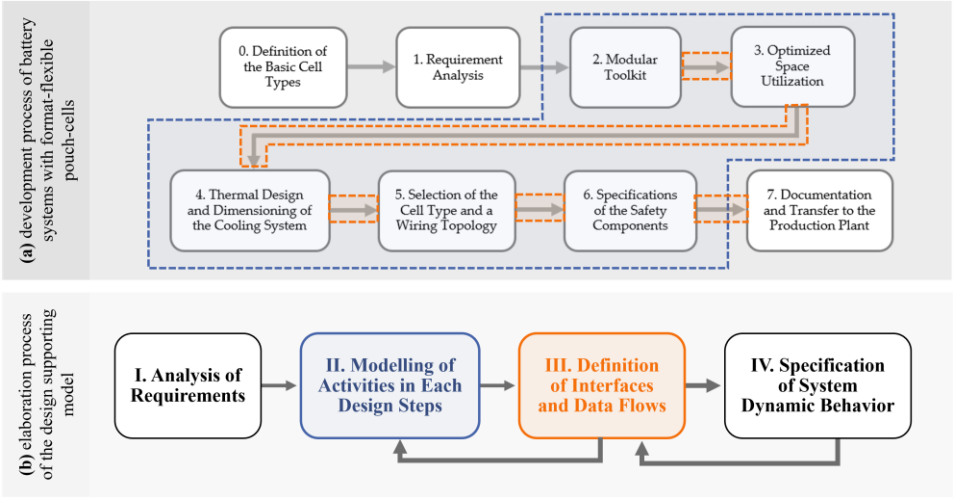

Figure 1: **(a)** The design process of battery systems with format-flexible pouch cells as presented in (Gandert et al. 2024) and **(b)** the elaboration process of the design supporting model; Context of step II (blue) and step III (orange) corresponding to the design process are marked accordingly.

The elaboration process of the methodology can be divided into the following steps with the enumeration corresponding to **Figure 1 (b)**:

I. **Analysis of Requirements**: This stage involves identifying the needs and expectations of the stakeholders, who are expected to work with the design supporting model. This provides the basis for determining what the system should accomplish and what features it should possess.
II. **Modelling of Activities in Each Design Steps**: Corresponding to the derived requirements, the activities involved in each design step, i.e. step 2 – 6 marked accordingly in **Figure 1 (a)**, are modeled, which includes the essential sub-activities and iterations.
III. **Definition of Interfaces and Data Flows:** In this step, interfaces between different system elements or activities are defined, along with the format and content of data flows between them. This involves identifying the inputs and outputs, demonstrating how data is exchanged and processed within the system.
IV. **Specification of System Model Dynamic Behavior:** This stage focuses on specifying how the system model behaves dynamically, based on the dependencies between different design steps, in response to various inputs and events initiated by the user. It includes defining sequential and concurrent behavior over time as well as state transitions.

2.1 Analysis of Requirements (Step I)

Following the approach presented in **Section 1.1** and **Figure 1 (a),** we conducted the development of battery systems with format-flexible pouch cells ourselves, during which we observed and consolidated the encountered difficulties. Moreover, we have identified challenges associated with collaborating with engineers from diverse domains during this process. Given these challenges, it is necessary to distinguish between different roles undertaken by people involved in the development process. There are engineering teams that are developing the battery system using the design tool and method described in **Figure 1 (a)**. Also, there are engineering teams that have developed or are improving the described tool and method. The developed and described system model in this article is aiming to consider both views and address the regarding challenges.

Due to the complexity of the design tools involved in the development of battery systems and the close interaction of the various domains, it is difficult for the engineering teams, which are respectively responsible for one of the design steps, to keep track of the preceding and subsequent steps as well as the interaction between them. The lack of the overview and information makes it challenging to proceed with the design process smoothly or make decisions effectively. Engineering teams may either initiate the design process and later discover the need for input from previous steps, or they may modify the design within their own domain without realizing the potential impact on subsequent steps. Furthermore, the absence of data traceability poses a challenge in understanding where data is utilized or modified throughout the process. It becomes uncertain whether the data structure at the end, altered through numerous manual changes, contains the latest design information. Considering the reusability of this approach, each step should be clearly documented and understandable as well.

The challenges described lead to several requirements:

- Overview and Sequence: The overall process flow, sequence of design steps and the iteration loops should be presented.
- Guidelines for User: This involves establishing guidelines to aid users in quickly assessing the current stage of development, providing input, understanding subsequent steps, and identifying where input may be lacking throughout the design process.
- Traceability of Data: To ensure clear traceability of data within the resulting dataset, it is essential to capture where data is utilized or modified within each design step and ascertain whether the data is adequate for initiating the next step.
- Comprehensive Documentation: There is a need for providing a description of individual sub-activities and sub-functions in the design steps to ensure clarity and understanding for each discipline involved.

2.2 Modelling of Activities in each Design Steps (Step II)

Among the options of MBSE to specify system behavior (activity diagram, sequence diagram and state machine diagram), we chose to implement the methodology using activity diagram, as it provides the best readability and is particularly good at expressing the flow of objects and complex control logic. The design steps 2 – 6 according to **Figure 1 (a)** are in the focus of our consideration and are represented as a distinct box within the diagram. Apart from the design steps, data preprocessing and data storage in form of MATLAB Struct are included as essential activities. An annotation panel is added, where description or information of individual sub-activities and sub-functions is provided. Within each design step, various activities shall be performed to accomplish specific objectives. These activities are represented as sub-nodes or components within the corresponding step box. The sequence in which activities are performed within each design step

as well as between them are then defined indicated using dash arrows. The activity diagram of the design supporting model is shown in **Figure 2,** illustrating the overview of modeled design steps and essential relating activities.

2.3 Definition of Interfaces and Data Flows (Step III)

Activities within the design steps that requires inputs and/or produces outputs are identified in the first place. The interfaces are represented using input/output pins attached to the activity box and the arrows between the pins indicate the flow direction. Some design steps involve decision points where alternative paths or branches can be taken based on certain criteria or user behavior. **Figure 2** shows some examples of the defined interfaces as well as the flow of data between activities.

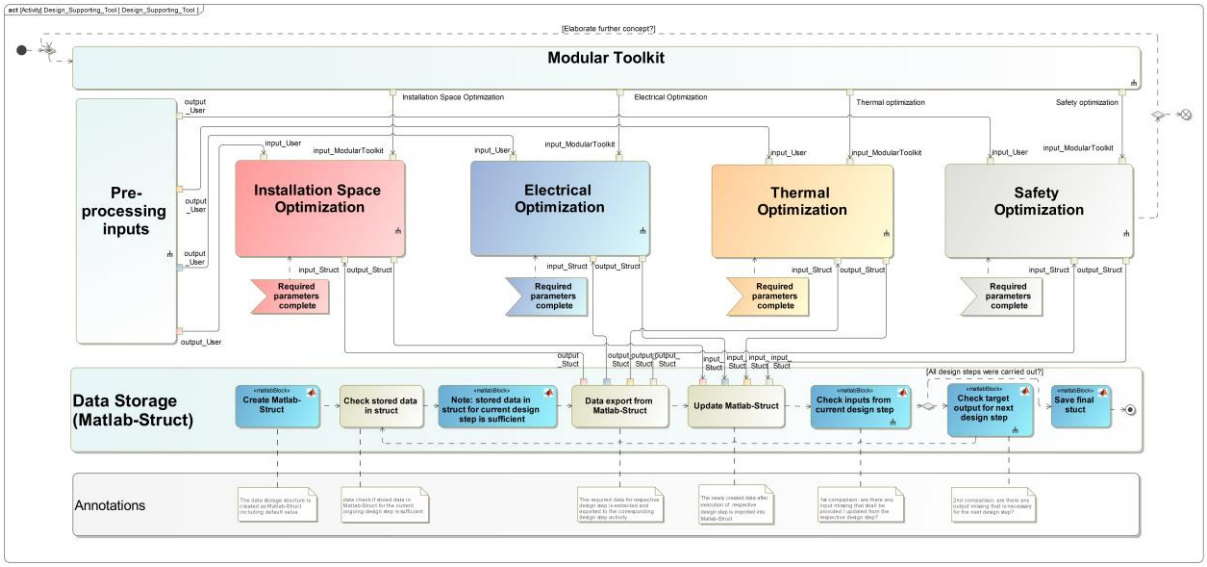

Figure 2: Overview of the modeled design steps and key activities using MBSE.

2.4 Specification of System Model Dynamic Behavior (Step IV)

To model asynchronous behavior of the system model, for instance when certain activity must wait for another event occurrence before it can continue its execution, signal event is used. The signal is set after the prerequisite activity, whose notation is a convex pentagon (shaped like a signpost). Once the prerequisite activity is carried out, this signal is triggered and will be accepted by the activity on the receiving end. The reception of a signal is indicated by a concave chevron arrow.

Within each activity box, it is possible to extend functionality by integrating scripts or functions from supported software. For example, the workflow can be enhanced by integrating certain MATLAB functions to execute specific tasks. We integrated MATLAB functions for comparing and verifying data, i.e. ensuring that the input data conforms to the required format and completeness before proceeding with subsequent steps as well as assessing that all relevant information of output generated by the design process is captured and stored appropriately.

To enhance usability, users are provided with relevant information through pop-up windows whenever input or decision-making is required.

The design supporting model is elaborated in an iterative approach, once new activities, functions or interfaces are added into the existing workflow, the overall process is executed to verifying the compatibility with existing functionalities and assessing whether the overall process achieves the desired outcomes.

3 Illustrative Application

The chosen application case of a battery system optimization using format-flexible pouch cells is a highly iterative process with multiple domains and interactions. It requires different individual optimization tools. These subtools are often developed separately with multiple inputs and outputs which have to be matched to other subtools. They are also partly automated with iterations of several parameters or need user inputs at some point. Therefore, this example offers challenges both for the developer of the optimization tools as well as the developer of the battery system who is using these tools. The developed methodology of this paper applied to one part for support.

3.1 Modelling of Installation Space Optimization (Step II)

An essential step in the design and optimization of a battery system with format-flexible cells is the filling of the installation space available for the battery system. This step is discussed in more detail in the following example. This consists of the essential steps of defining the installation space itself. The module configuration with its components consisting of cells, cooling and safety components. The installation space is then scanned to determine the possible module and cell sizes. In addition, different definitions of the module configuration or arrangement orientation can be evaluated by iterating the procedure in order to find an ideal arrangement of the cells in the installation space. The exact procedure is described in (Müller-Welt et al., 2022). The individual functions are modelled as actions within the main activity of installation space optimization. An overview of this can be seen in **Figure 3**.

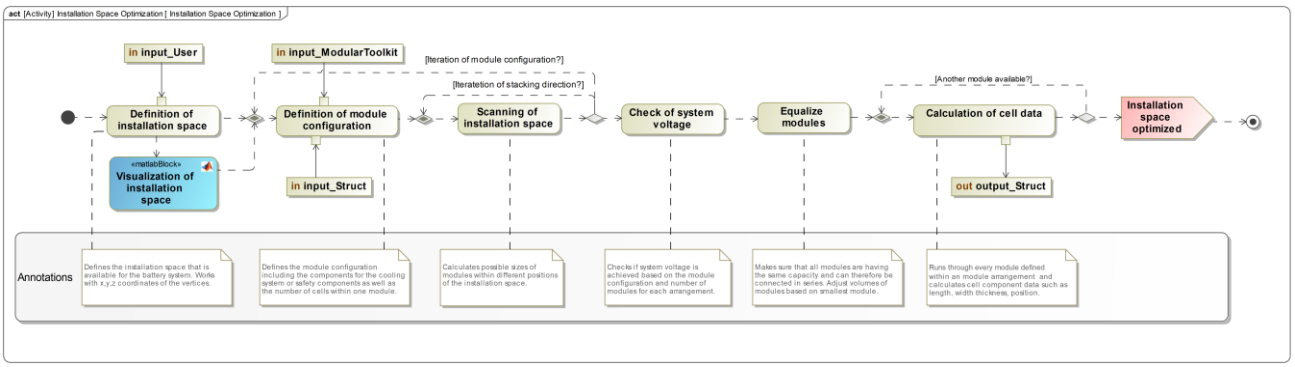

Figure 3: Main activities within the installation space optimization activity

The individual actions shown represent functions within the design supporting model and in turn consist of sub-functions which, for example, realize the geometric determination of the maximum dimensions of the cells that can be integrated into the installation space or the determination of the layer structure required for the respective cells. In addition, iterations were implemented using the varied parameters such as cell thickness, cell type, arrangement direction and module configuration by means of feedback from the control flow in the activity diagram. These are also required as input for the execution of the design step. Some of these can either be defined directly by the user or are transferred from other design steps.

This type of mapping of the individual functions provides the user of the design supporting model with graphical support for understanding the overall process that is used to optimize the cell configuration within the installation space as well as the nested program sequence. The implementation as an activity diagram also provides a simple way of examining the program flow when integrating new functions or further iteration loops or case distinctions.

3.2 Definition of Interfaces and Data Flows (Step III)

A very important factor in ensuring the functionality of the entire design supporting model is the definition of interfaces for data transfer between the individual subsections. In the case of installation space optimization, interfaces to the preceding modular toolkit, the user and the shared data structure are provided for this purpose. These are shown as examples in **Figure 4 (a)**.

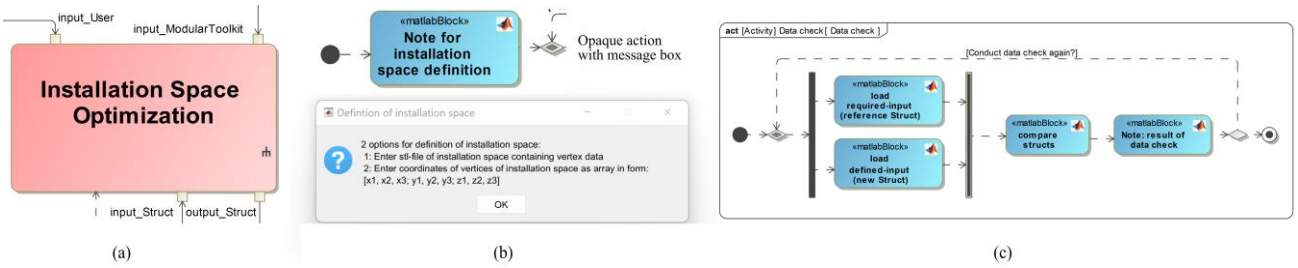

Figure 4: **(a)** Definition of interfaces or step of installation space optimization, **(b)** additional information for user input, **(c)** implementation of data check for completeness

To carry out the installation space optimization, the concept selected in the modular system is first required, which largely defines the module design and the installation space required for additional components for the cooling system and safety aspects. A fixed data format was defined for these, which was modelled as the required input "input_ModularToolkit" for the installation space optimization. The required parameters were also defined by boundary conditions from production, such as maximum and minimum cell dimensions or available cell types, consisting of a specific material and properties of

the cell layers, and an "input_Struct" data structure was defined for the transfer. This is read from a higher-level data struct, which is used for the entire design process. Similarly, the "output_Struct" interface was modelled to transfer the results of the installation space optimization. Finally, there is also an interface for input variables that must be defined by the user for the specific application "input_User".

To support the user, pop-up windows have also been modeled with detailed explanations of required inputs and case distinctions that appear during the execution of the activity diagram. An example of this can be seen in **Figure 4 (b)**. For example, there are two options for defining the available installation space. This is represented in the design supporting model by coordinates of the vertices. These can either be entered directly by the user in the form of x, y, z pairings or defined by importing a stl-file containing this data.

Another functionality for ensuring consistent data transfer in the modeled interfaces is the verification of data structures. This is particularly important for data transfer between design steps that are not developed or executed by the same person. In the case of installation space optimization, this primarily involves reading in data from the MATLAB struct on the part of the production-related boundary conditions and existing cell types, as well as transferring the data to the subsequent electrical and thermal design steps. For this purpose, actions were also modeled in the control flow before and after the respective interfaces, which enable a comparison of data structures. These are shown as examples in **Figure 4 (c)**. For the comparison, a required struct can always be compared with a received struct in order to quickly identify missing parameters.

3.3 Execution of Model for Support of Design Tool Application (Step IV)

The execution of the activity diagram in the MBSE model is partially illustrated using the example of the main activities of the installation space optimization in **Figure 5**. This shows a visualization of the program flow. This can be used in the application for various aspects. On the one hand, it can be used to test the integration of new functions or design sequences in the overall process. On the other hand, this supports users of the design supporting model who were not directly involved in the development. This can be particularly helpful in the many interactions between the definition of a cooling concept, the subsequent filling of the installation space, the thermal and electrical evaluation and the potential redefinition of the cooling concept.

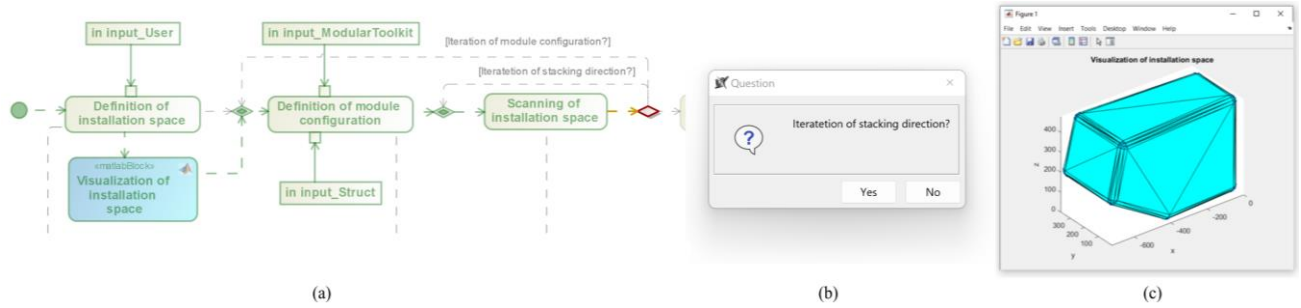

Figure 5: Illustrative execution of activity for installation space optimization

The process is run through step by step and highlighted in color as shown in **Figure 5 (a)**. Green parts represent steps that have already been run through, yellow parts represent steps that have already been run through and red parts represent the current step. This visual support is particularly helpful in the context of the overall process, as it makes it possible to quickly identify program sections that have not been run through in the event of case distinctions or causes for aborts in the event of missing inputs. Further assistance could be provided in the form of queries to the user during iterations or case distinctions as well as direct integration of partial results from MATLAB functions into the workflow. As an example, **Figure 5 (c)** shows a plot of the defined installation space, which is generated by calling the MATLAB function.

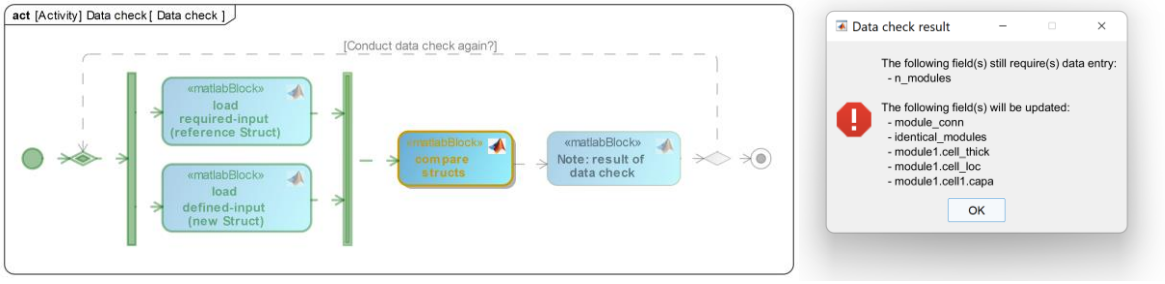

Figure 6: Illustrative data check for the resulting structure generated by filling the installation space

The resulting structure generated by filling the installation space can finally be compared with the structure required for the next design step. An example of how the function shown in **Figure 4 (c)** is used is shown in **Figure 6**. A key component is the identification of differences between the required and defined struct. After executing the function to compare these two data structures, the user receives information in the form of a pop-up window about where an entry is still missing, and which fields are updated with the result of the executed design step. The user can then specifically check the fields detected, modify them and run the function of data check again.

4 Discussion and Outlook

In this paper, a design supporting model is elaborated and demonstrated by applying MBSE to support the process of optimization of battery systems with format-flexible cells. An overview of the entire design process and associated functions is achieved, which provides users with a comprehensive understanding of the workflow, highlighting key activities and decision points. Annotations are included to provide explanations for understanding the design process. Some activities are extended to incorporate MATLAB functions within the design supporting model, in particular enable comparing and verifying the input data for each design step and output data produced afterwards. Upon completion of the design process, the final MATLAB struct containing relevant design properties is saved. By integrating MATLAB structures directly into the design supporting model, traceability and consistency of design information are maintained, facilitating collaboration and decision-making across the development cycle.

In an illustrative application, the successive activities during the design process were successfully mapped. In some cases, functionalities were implemented that provide support both during the development of the design supporting model and during its actual use. These include graphical representations of overall processes and notes on individual components, as well as functions to ensure the consistency of data sets that are exchanged between individual design aspects. These supports were developed based on requirements derived from a battery optimization example, which is a highly complex optimization process. The implemented design steps as well as interfaces are specific to this process. Nevertheless, the developed method can be applied to other development processes. It is especially offering benefits for iterative and multi-domain development processes.

The design supporting model is capable of fulfilling the initial requirements defined in **Chapter 2.1**. It provides an overview of the design process flow, as well as the involved design steps and design tools. Moreover, users of the design supporting model receive guidance through prompts that assist them in inputting relevant data and making design decisions. For the users of the design tools, i.e. a battery system developer, the benefits of our approach lie in streamlining the design process and facilitating decision-making. The design supporting model guides users through a structured workflow, prompts them to input relevant data at each step, which ensures that the user provides necessary inputs to better assess the current stage of development, identify areas for improvement, and understand the implications of design choices. As for engineering teams who work on developing specific design tools, i.e. the tool developers, the design supporting model facilitates comprehension of the interactions among the involved design tools, i.e. the required data from preceding steps and impact of output on subsequent ones, thus enabling them to design interface between design tools better and refine format of data exchange. In terms of traceability of data, the design supporting model captures where data is utilized or modified in each design tool and verifying its adequacy for subsequent steps. Lastly, annotations are provided, offering detailed descriptions of sub-tools and sub-functions.

In future work, the use of the activity diagram can be extended by feeding the results from the design tools back into the design supporting model and saving them. This can then be done in the form of block definition diagrams and instances for the entire system and thus be used for the further development process, e.g. for validation activities.

Acknowledgments

This work was funded by the Baden-Württemberg Ministry of Science, Research and the Arts within the project “AgiloBat” (reference number: 32-7533-4-161.2/9/12) as part of the Innovation Campus Mobility of the Future.

References

Albers, A., Lanza, G., Klippert, M., Schäfer, L., Frey, A., Hellweg, F., Müller-Welt, P., Schöck, M., Krahe, C., Nowoseltschenko, K., Rapp, S. (Eds.), 2022. Product-Production-CoDesign: An Approach on Integrated Product and Production Engineering Across Generations and Life Cycles. Procedia 32nd CIRP Design Conference, 7 pp.

Albers, A., Wagner, D., Hammami, W., Spadinger, M., Höfler, T., 2014. Integration of design methods for the development of electric energy storage systems, 10 pp. (accessed 19 March 2020).

Bouvy, C., Ginsberg, S., Jeck, P., Hartmann, B., Baltzer, S., Eckstein, L., 2012. Holistic Battery Pack Design. 21. Aachener Kolloquium 12.

Delligatti, L., 2014. SysML distilled: A brief guide to the systems modeling language. Addison-Wesley, Upper Saddle River, NJ, 1 online resource (1 volume).

Friedenthal, S., Moore, A., Steiner, R., Lockheed Martin, The MathWorks, Raytheon, 2012. A practical Guide to SysML. Morgan Kaufmann.

Gandert, J.C., Mühlpfort, R., Müller-Welt, P., Schall, D., Schmidt, A., Schuhmann, S., Seegert, P., Weber, N., Zeng, Y., Wetzel, T., Paarmann, S. Methodology for the Holistic Design of Format-Flexible Lithium-Ion Battery Systems, vol. 2024, pp. 1–17.

Lensch-Franzen, C., Gohl, M., Schmalz, M., Doguer, T., 2020. Von der Zelle zum Batteriesystem - Verschiedene Zellformate und ihre Systemintegration, in: MTZ Motortech (MTZ - Motortechnische Zeitschrift), vol. 81, pp. 70–75.

Müller-Welt, P., Bause, K., Albers, A., 2023. Battery System Optimization Using Format Flexible Pouch Cells – Literature Review and Research Methodology, in: Procedia CIRP, vol. 119, pp. 1078–1083.

Müller-Welt, P., Nowoseltschenko, K., Garot, C., Bause, K., Albers, A., 2022. Automated Optimization of a Cell Assembly Using Format-Flexibly Produced Pouch Cells, in: 22. Internationales Stuttgarter Symposium Proceedings, pp. 569–581.

Rajasekhar, M.V., Gorre, P., 2015. High voltage battery pack design for hybrid electric vehicles, in: 2015 IEEE International Transportation Electrification Conference (ITEC), pp. 1–7.

Ruhland, J., Storz, T., Kößler, F., Ebel, A., Sawodny, J., Hillenbrand, J., Gönnheimer, P., Overbeck, L., Lanza, G., Hagen, M., Tübke, J., Gandert, J., Paarmann, S., Wetzel, T., Mohacsi, J., Altvater, A., Spiegel, S., Klemens, J., Scharfer, P., Schabel, W., Nowoseltschenko, K., Müller-Welt, P., Bause, K., Albers, A., Schall, D., Grün, T., Hiller, M., Schmidt, A., Weber, A., Biasi, L. de, Ehrenberg, H., Fleischer, J., 2022. Development of a Parallel Product-Production Co-design for an Agile Battery Cell Production System, in: Towards Sustainable Customization: Bridging Smart Products and Manufacturing Systems, vol. 9, pp. 96–104.

Schmalz, M., Lensch-Franzen, C., Kronstedt, M., Wittemann, M., 2020. Skalenübergreifende Batteriesystembetrachtung zur effizienten Entwicklung und Optimierung elektrischer Antriebe, in: Liebl, J. (Ed.), Experten-Forum Powertrain: Simulation und Test 2019, vol. 10. Springer Fachmedien Wiesbaden, Wiesbaden, pp. 15–24.

Stachowiak, H., 1973. Allgemeine Modelltheorie. Springer Verlag, Wien.

Walden, D.D., Roedler, G.J., Forsberg, K., Hamelin, R.D., Shortell, T.M., Engineering, I.C.o.S. (Eds.), 2015. Systems engineering handbook: A guide for system life cycle processes and activities, 4th ed. Wiley, Hoboken, New Jersey, 290 pp.

Weilkiens, T., 2006. Systems engineering mit SysML/UML: Modellierung, Analyse, Design, 1st ed. Dpunkt-Verl., Heidelberg, 360 pp.

Contact: Yunying Zeng, Karlsruhe Institute of Technology (KIT), IPEK – Institute of Product Engineering, Kaiserstraße 10, 76131 Karlsruhe, Germany, +49 721 608-47200, +49 721 608-45752, yunying.zeng@kit.edu

Contact: Philip Müller-Welt, Karlsruhe Institute of Technology (KIT), IPEK – Institute of Product Engineering, Kaiserstraße 10, 76131 Karlsruhe, Germany, +49 721 608-47254, +49 721 608-45752, philip.mueller-welt@kit.edu

Contact: Katharina Bause, Karlsruhe Institute of Technology (KIT), IPEK – Institute of Product Engineering, Kaiserstraße 10, 76131 Karlsruhe, Germany, +49 721 608-46992, +49 721 608-46966, katharina.bause@kit.edu