Assistance System for graph-based 3D Visualization of Design Structure Matrices

Jan-Phillip Herrmann[1], Sven Tackenberg[1], Christoph Trojanowski[2], Carolin Pankrath[3], Sebastian Imort[1], Andreas Deuter[1]

[1]OWL University of Applied Sciences and Arts, Lemgo, Germany
[2]University of Applied Sciences, Berlin, Germany
[3]LMtec Service GmbH, Potsdam, Germany

Abstract: The increasing number of product artifacts (e.g., mechanical or electronic components, software functions, documents) confronts small and medium-sized companies with the challenge of assessing change effects. The lack of knowledge of artifact relationships causes problems, such as outdated documentation, lack of coordination with affected disciplines, or delayed changes. The Design Structure Matrix (DSM) can clearly represent the elements and relationships of complex systems. This paper presents an assistance system for intuitive visualization of engineering change effects using existing DSM-based methods for complexity management. The implemented algorithms compute graph layouts, cluster analyses, and change predictions in the form of change risk, time, and cost. An application example of a 3D-printed intelligent lamp demonstrates the approach's viability. The paper concludes with a discussion of the benefits and future activities.

Keywords: Graph-based Visualization, Assistance System, Engineering Change Management, Complexity Management

1 Introduction

As a result of the ongoing digital transformation, small and medium-sized enterprises (SMEs) face increasing complexity in developing and managing cyber-physical systems across the entire product lifecycle. Technological change, product variety, and process agility are complexity drivers that significantly impact product development goals in terms of time, cost, quality, and flexibility (Latos et al., 2018; Vogel and Lasch, 2016). Unknown dependencies among product development artifacts have been identified as one of the core problems SMEs face in managing complexity during product development. In contrast to large companies with dedicated systems engineering departments, SMEs need more resources to document and manage artifact relationships thoroughly. Accordingly, information about such relationships is mostly tacit knowledge of product developers. The more complex products become, the less the relationships between artifacts are known to the stakeholders involved. With the existence and retrieval of such decision knowledge, the effects of technical changes on other artifacts can be predicted.

The Design Structure Matrix (DSM) is a powerful method for modeling complex technical systems and can represent a wide range of information on the structural complexity of products and processes (Browning, 2016; Sinha and de Weck, 2013; Eppinger and Browning, 2012). Contrary to the creation of detailed system models (e.g., Jacobs et al., 2022), the authors of this paper believe that the creation of SysML models for SMEs is too time-consuming and requires an in-depth expert knowledge of the system and the modeling language. Therefore, we use a binary DSM as a cost / effort-efficient description of a complex system for SMEs. The German research project "Function-oriented complexity management in all phases of the product development process" (German acronym: FuPEP) (see acknowledgments) develops an assistance system for SMEs that supports product developers in managing complexity during the development process. In particular, the assistance system aims to help product developers create an awareness of the relations between relevant product artifacts and reflect on the consequences of technical changes. This paper presents an assistance system for intuitive visualization of engineering change effects using existing DSM-based methods for complexity management. Section 2 introduces the related work for assistance systems for complexity management. Based on a previous requirements elicitation for an assistance system described in Section 3, the paper presents the software architecture and the adapted algorithms for assistance-supported complexity management. We introduce the graphical user interface, which has been realized using a gaming engine. This interface uses DSM-based clustering and engineering change management functions to generate the results for the visualization. The developed assistance system addresses product and process developers of SMEs. To illustrate the DSM-based methods used and the benefits of algorithm-driven complexity management, a 3D-printed intelligent lamp ("SmartLight") serves as a use case. Finally, the paper concludes in Section 4 with a discussion of the benefits and limitations of the assistance system.

2 Related Work

Previous studies particularly relate to the visualization of artifact relationships and the comprehensibility of graph-based representations of matrices. Tools for complexity management, like Siemens' PLM software, Teamcenter (Herbst and Hoffmann, 2018), offer various functionalities to assist product developers during initial design and revision by

establishing relationships between different artifacts. However, there is a need to be more intuitive in visualizing and analyzing the propagation of changes in complex systems to be more efficient.

The intuitively understandable representation between artifacts is a success factor for complexity management. In the literature, empirical studies exist on DSM-based and graph-based forms of visualization. Jarratt et al. (2004) describe a link connection plot, which is also called a Molecular diagram (Sharman and Yassine, 2004) that depicts system elements as labeled nodes and relationships as directed edges. Peterson (2015) utilizes DSMs and network analysis to organize and cluster complex systems into graphical representations. Tools like Gephi (open source) (Bastian et al., 2009) and Pajek (Batagelj and Mrvar, 2004) offer network analysis and visualization capabilities for free non-commercial use. Ghoniem et al. (2005) and Keller et al. (2006) compare the readability of design structure matrices and their representation as a random 2D graph. They define several tasks, like counting the number of incoming or outgoing links or finding a path between two nodes. While participants perform these tasks on over 100 graphs with different sizes and densities, the response times and error rates are measured. Although the DSM outperforms the graph representation in most tasks, both studies confirm that participants perform better on graph representations when the task is related to identifying paths and connectivity in the graph topology. This is also confirmed by more recent studies (Okoe et al., 2018). Vessey et al. (1991) present a cognitive fit theory between the representation and task types. They compare human information processing performance between tabular and graph-based information representations. While tables emphasize symbolic information, graphs provide spatial information. Identifying relationships and paths between elements is a spatial task that requires graphs as a spatial representation.

3 The Assistance System

3.1 Requirements and Architecture

The present work aims to develop a concept for visualizing the effects of product and process changes using existing DSM approaches. The result is an assistance system with a graphical user interface for the complexity management of SMEs. The requirements for such an assistance system were surveyed in two German SMEs. The authors collected and prioritized 30 use cases and 130 assistance system requirements in twelve individual and two group interviews using a structured questionnaire. Table 1 provides an overview of the use cases rated as particularly relevant by the product developers for using the assistance system.

Table 1. Most important use cases

Use Case	Description	In-Scope?
UC001	Making dependencies between product development artifacts visible	YES
UC008.a	Supporting system function approvals and their communication	NO
UC012.a	Artifact information searching capabilities	YES
UC012.b	Intuitive graphical information representation	YES
UC012.f	Representation of engineering change impacts	YES

Requirement UC001 describes the intuitively understandable representation of relationships between artifacts from different domains. For example, an artifact represents a function, a document, a product requirement, a physical part, or a test case. Such a representation helps to analyze the impact and propagation of changes on other development artifacts. Especially, artifacts can be indirectly related to each other through intermediate artifacts. Changes can propagate across intermediate artifacts. With growing system complexity, product developers are decreasingly aware of these indirectly affected artifacts and benefit from graphical support for their identification. Requirements UC012.a, UC012.b, and UC012.f refer to information about artifacts and relations. According to the requirements, developers want to visualize the impact of technical changes on other artifacts. Hence, the artifacts affected by a change to an artifact are to be automatically identified and displayed to the user. According to the respondents in the companies, it is not necessary to describe the specific characteristics of a change impact in detail. Thus, it is sufficient if, for example, the diameter of a borehole is changed, and the affected asset thread is displayed to the user. Furthermore, it is not necessary to specify the modified geometry of the thread, as existing software tools cover this function. The Scope column indicates whether the assistance system's development stage fulfills the individual requirement. A detailed list and description of all use cases and requirements elicited was published by Herrmann et al. (2021).

Figure 1 shows the three-tier assistance system architecture addressing the use cases and requirements. An asset administration shell holds all artifact and relationship-related information in the assistance system. It is a standardized digital representation of assets and uses submodels, submodel collections, and elements to describe artifacts in terms of attributes and relationships (Plattform Industrie 4.0, 2023). The Asset Administration Shell Server (b) imports artifacts and their relationships from third-party systems like ERP or PLM systems (a) via a Data Manager and stores them into

asset administration shells. On receiving requests from the graphical user interface (d), the DSM server retrieves a DSM object containing all relevant information for computing the algorithms and visualizations from the Asset Administration Shell Server via HTTP/REST. After executing the algorithms on the DSM object, the DSM server (c) sends the DSM object via google Remote Procedure Call (gRPC, 2023) to the graphical user interface. The graphical user interface is implemented in Unity, a cross-platform game engine for developing games (Unity Technologies, 2023).

The current implementation generates the asset administration shell from an Excel file since the automated import of artifacts and relationship information is still in development. Therefore, the user must manually specify all artifacts and relationship information in the Excel file. For a detailed description of the asset administration shell representation of a DSM and the Excel file structure, the reader is referred to Imort et al. (2022).

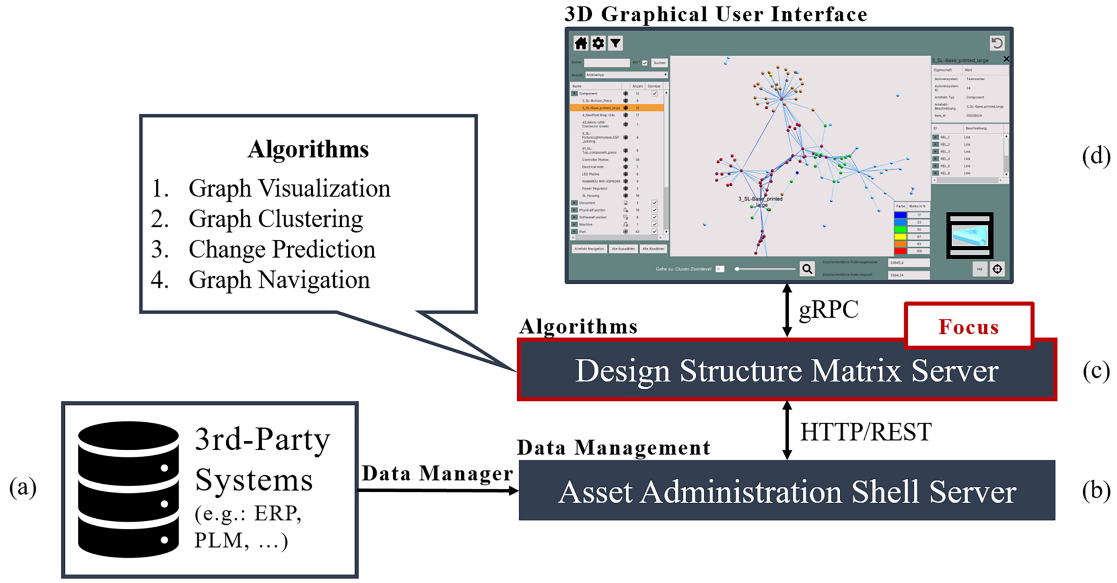
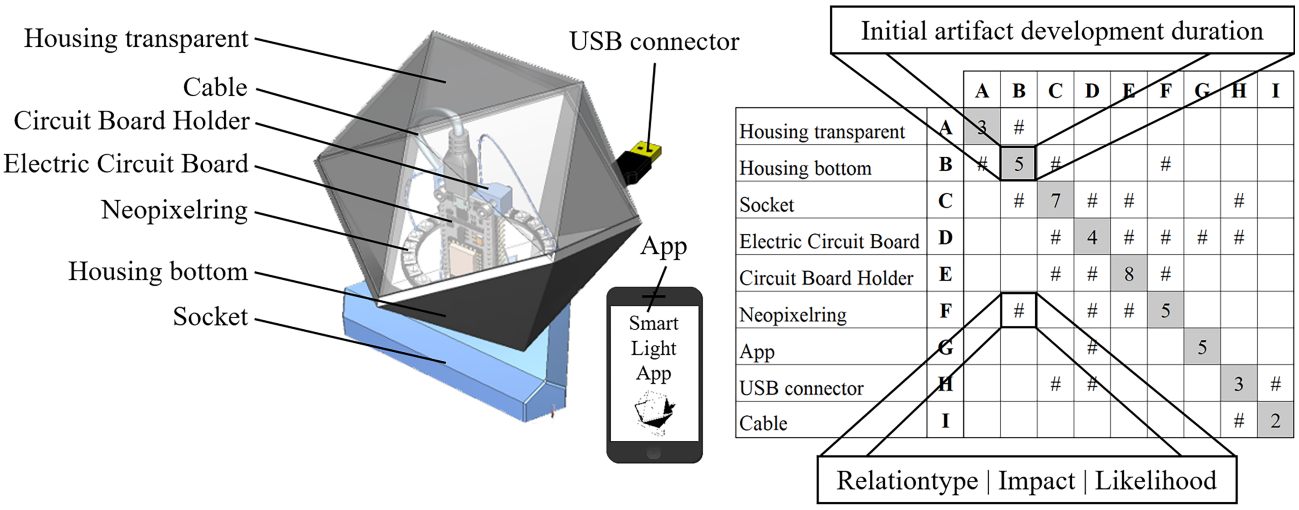


Figure 1. Three-tier assistance system architecture

3.2 Example Application – SmartLight

An application example of a LED lamp with a 3D-printed housing that can be controlled via a desktop app is used to develop and evaluate the functionalities of the assistance system. The app connects to the electric circuit board mounted on the SmartLight through a WIFI module. The app lets users turn the light on and off and change the light color.

The SmartLight approximates a cyber-physical system that integrates artifacts from different domains like physical parts, electronics, and software (Figure 2, left). The assistance system focuses on all product development artifacts and is not restricted to a specific domain, resulting in domain-independent multi-domain matrices. However, the abbreviation "DSM" will be used in the remainder of this paper to stay consistent.



		A	B	C	D	E	F	G	H	I
Housing transparent	A	3	#							
Housing bottom	B	#	5	#			#			
Socket	C		#	7	#	#			#	
Electric Circuit Board	D			#	4	#	#	#	#	
Circuit Board Holder	E			#	#	8	#			
Neopixelring	F		#		#	#	5			
App	G				#			5		
USB connector	H			#	#				3	#
Cable	I								#	2

Figure 2. Example application – SmartLight

3.3 Graph Visualization and User Interface

The graph-based representation of artifact relationships has advantages over a matrix-based form regarding comprehensibility. Thus, Keller et al. (2006) demonstrated in an empirical study that subjects showed significantly lower response times and error rates when finding links or analyzing shortest paths in a graph than in a DSM. Therefore, our assistance system employs graph drawing algorithms in three-dimensional space, and extending the graph layout to three dimensions increases the space of possible node positions. This results in reduced edge crossings, enhancing graph aesthetics (Herman et al., 2000; Purchase, 1997) so that improved comprehensibility is assured.

The current assistance system's implementation focuses on binary symmetrical DSMs drawn as undirected cyclic graphs. A generalization of the ForceAtlas2 algorithm to three-dimensional space illustrates a DSM as a graph. This graph layout algorithm is used in the graph visualization software Gephi (Jacomy et al., 2014). Figure 3 shows the assistance system's automatically generated main view when a user opens the asset administration shell of a DSM. On the left, users can select artifacts in a tree view. The tree view displays the hierarchy of artifacts according to their artifact type (e.g., component or process). Also, the tree view allows users to hide and show artifact groups of interest. The ForceAtlas2 algorithm calculates the graph layout considering the Scaling Ratio and Gravity parameters. Figure 4 shows the stepwise convergence of the ForceAtlas2-generated graph layout to a local solution.

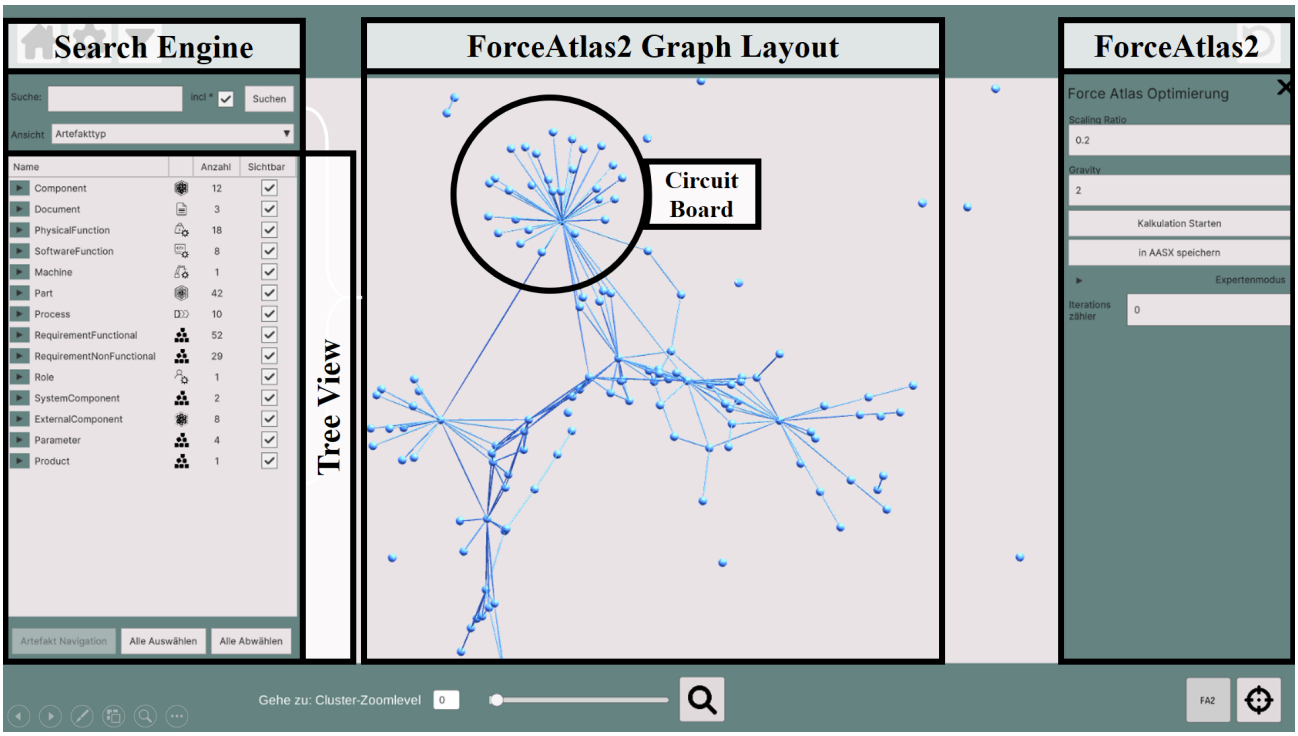


Figure 3. Main view in the assistance system's user interface

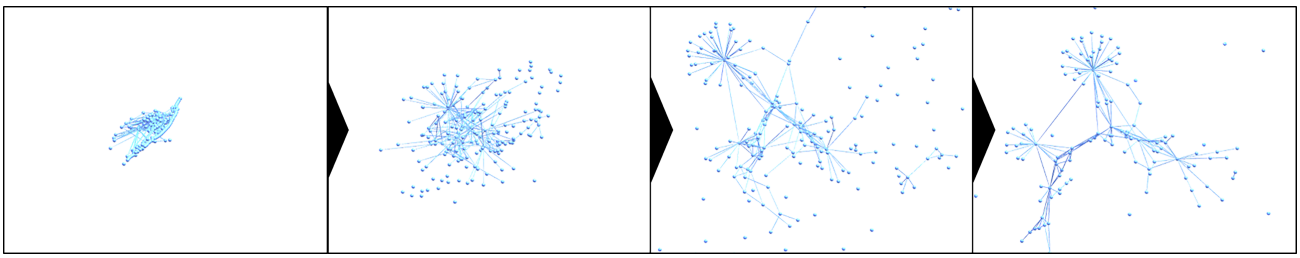


Figure 4. Evolution of the ForceAtlas2 algorithm over 5,000 iterations (approximately)

On mouse hover, nodes are labeled by their artifact name. The graph layout provides users with an overview of the general architecture of the considered system. E.g., the architecture of the SmartLight's circuit board is given in the top left-hand corner of the graph layout (see Figure 3, center). The circuit board has geometric, informational, and energetic relationships with the other SmartLight artifacts, such as the housing, the app, or the light ring. On the other hand, many electronic components directly relate to the circuit board, resulting in a star-like structure. From these observations, users can visually

assess the modularity of products and their artifacts. E.g., artifacts organized modularly are drawn as chain-like structures. Highly bus-modular systems like the electronic circuit board exhibit star-like structures.

A property window is displayed when clicking on an artifact in the tree view or on its respective sphere in the graph layout. This window contains information describing the artifact, e.g., its name, identification number, origin (third-party system), and direct relationships to other artifacts. Figure 5 (right) illustrates the property window when the user selects the artifact ["3_SL-Base_printed_large"]. A navigation feature allows users to fly through the three-dimensional space. Therefore, users can analyze the details of the graph structure, specific artifacts, and their relationships.

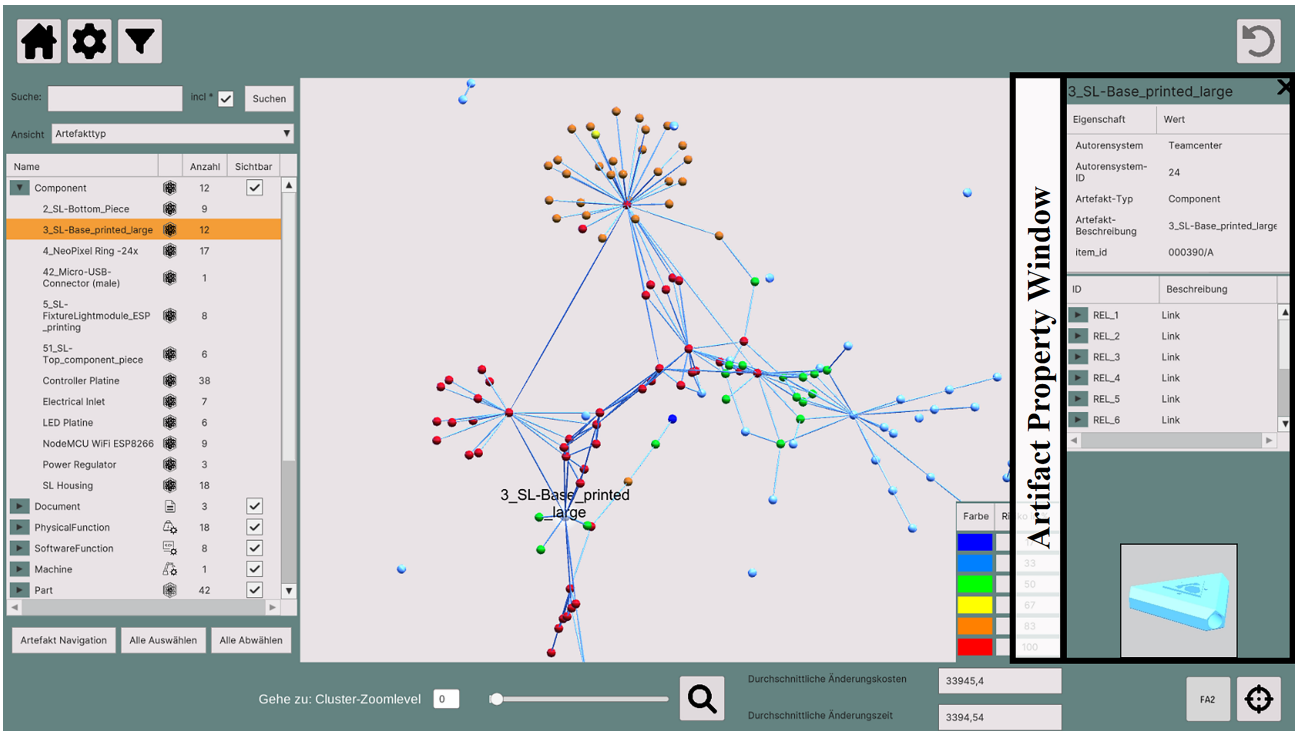


Figure 5. Selection of artifacts

3.4 Graph Clustering

Graph clustering functionalities automatically simplify the presented graph layout and identify logical modules and buses in the DSM. Logical modules represent an aggregation of several interrelated artifacts, and buses are artifacts that act as carriers of others (Sharman and Yassine, 2004). Various clustering algorithms for DSMs have been developed in the past (Yu et al., 2007; Idicula, 1995; Borjesson and Hölttä-Otto, 2013). For clustering the ForceAtlas2-generated graph layout, the assistance system employs the k-means algorithm (Lloyd, 1982; MacQueen, 1967). Input to this algorithm is the artifacts' adjacency matrix resulting from the graph layout positions. In this manner, clustering is performed based on the graph structure the user is provided with and shall help form intuitive clusters. Furthermore, the k-means algorithm allows generating clusters for large DSMs efficiently when performed with Lloyd's algorithm due to its linear time complexity. Figure 6 shows the pseudocode and the resulting ForceAtlas2 graph layout after the DSM clustering.

Algorithm 1 computes the ForceAtlas2 for each component of the DSM's graph. The affinity matrix is calculated based on the generated artifact positions in three-dimensional space. The affinity matrix is input to the k-means algorithm. Subsequently, the optimal number of clusters k is determined using the Elbow Method (see Cui (2020) for an introduction to the k-means algorithm using the Elbow Method). This method performs the k-means algorithm for all admissible numbers of clusters k. The number of clusters k with the lowest distortion (a quality criterion of the clustering) is selected as the optimum. Finally, the algorithm reorders the initial DSM's rows and columns by locating artifacts within a cluster next to each other. However, rows and columns belonging to a cluster remain unordered.

Using the results from Algorithm 1, Algorithm 2 condenses the initial DSM by generating a new DSM. In contrast to classification, clustering does not have predefined cluster labels. Therefore, for each cluster, a representative element name is determined using the PageRank algorithm (Page et al., 1999), namely, the element with the highest rank. Next, Algorithm 2 creates a condensed DSM with all representative elements in its rows and columns. Instead of binary values, relationships store the number of relationships within and between clusters from the initial DSM. Figure 6 (right) shows the resulting graph layout where each sphere represents a cluster. The spheres are named according to their corresponding cluster's representative element. Algorithms 1 and 2 are implemented in Python using the packages Scikit-learn (2011)

and Networkx (Hagberg et al., 2008). Both algorithms allow product developers to form logical modules of artifacts from different domains and analyze the resulting architecture.

Algorithm 1 Cluster Analysis

Require: dsm
  $dsm \leftarrow Reorder(dsm, randomOrder)$
  $components \leftarrow GetConnectedComponents(dsm)$
  for all $co \in components \wedge Order(co) > 1$ **do**
    $positions \leftarrow ComputeForceAtlas2(co)$
    $m_{Aff} \leftarrow ComputeAffinityMatrix(positions)$
    $n_{opt} \leftarrow ComputeOptimalNumberOfClustersUsingElbowMethod(m_{Aff})$
    $clustering \leftarrow KMeansAlgorithm(n_{opt}, m_{Aff})$
    for all $cluster \in clustering$ **do**
      $clusters \leftarrow clusters \cup cluster$
    end for
  end for
  $dsm \leftarrow Reorder(dsm, clusters)$
  return dsm

Algorithm 2 Condense DSM

Require: $dsm, clusters$
  $dsm_{clustered} = \emptyset$
  $representativeElements = \emptyset$
  for all $cl \in clusters \wedge Order(cl) > 1$ **do**
    $reprElement \leftarrow GetFirstElementInPageRank(cl)$
    $representativeElements \leftarrow representativeElements \cup reprElement$
  end for
  for all $row \in representativeElements$ **do**
    for all $column \in representativeElements$ **do**
      $noOfRel \leftarrow GetNoOfRelationshipsBetween(row, column)$
      $dsm_{clustered} \leftarrow dsm_{clustered} \cup (row, column, noOfRel)$
    end for
  end for
  return $dsm_{clustered}$

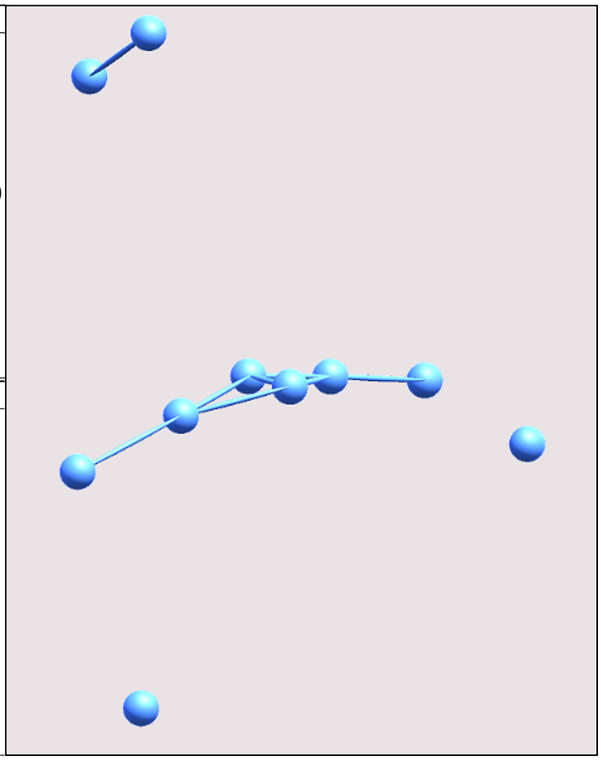


Figure 6. Pseudocode and resulting graph layout of a cluster analysis

3.5 Change Prediction

The assistance system implements functionalities for automatically predicting the propagation and effects of engineering changes through the system. It uses an adapted version of the Change Prediction Method (CPM) in Clarkson et al. (2004), combining the likelihood and impact of changes on other components into a measure of risk:

1. The CPM generates a propagation tree of all simple paths (paths without cycles) between a specified start and target elements.

2. The combined likelihood of change propagation between the start and target element is computed based on the propagation tree.

3. The combined risk is computed, incorporating the impacts into the combined likelihood computation.

The reader is referred to Clarkson et al. (2004) for a detailed method description. Calculating change risk from one artifact to another requires information on the propagation likelihood and change impact. For the CPM, users must provide two numerical DSMs with the likelihood and impact of changes as entries. The user has to specify these two numerical DSMs in the Excel file described in Section 3.1. The assistance system's adapted version of the CPM additionally allows considering cycles in the propagation tree. Furthermore, it enables defining an arbitrary amount of target elements instead of one. When changing artifacts, users can compute the effects on the whole or subset of the system.

If the change risk on a user-specified artifact is to be determined, the corresponding graph node is colored. The calculated CPM output can take values between zero and one and is displayed in a color scheme from blue to red. Figure 7 illustrates a risk calculation for changing the component ["4_NeoPixel Ring -24x"] and the color scheme for different risk values. The assistance system automatically colors all affected system elements when clicking on a node. A user who initiates a change receives an overview of the expected effects. Accordingly, the user can communicate the necessary information to the relevant actors. This ensures that the effects of changes are quickly identified, and users can implement corrective actions to minimize the required effort. Figure 8 provides the pseudocode for computing the risk of an engineering change on the remaining system.

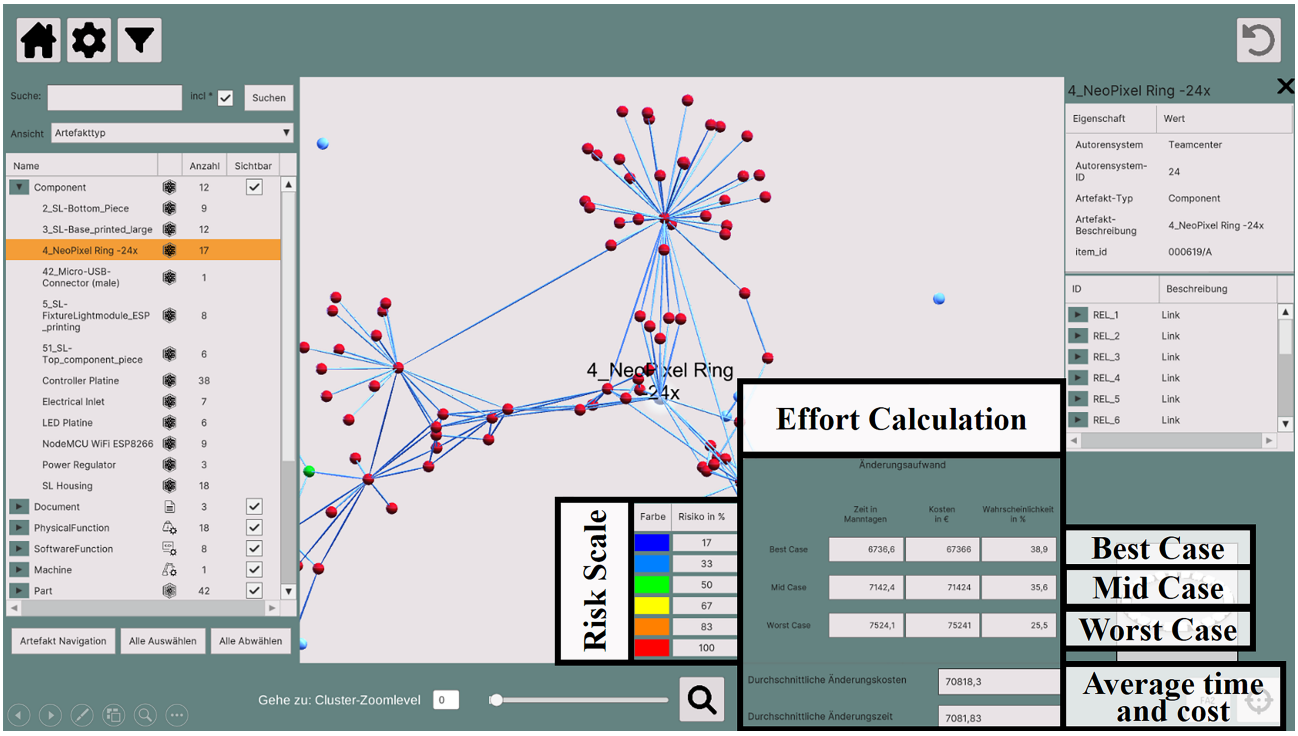


Figure 7. Change Prediction based on node coloring and change time and cost calculation

Algorithm 3 Change Prediction

Require: $dsm, dsm_L, dsm_I, allow\_cycles$
  for all $n_{Source} \in dsm$ **do**
    for all $n_{Target} \in dsm$ **do**
      $column_R \leftarrow ChangePredictionMethod(n_{Source}, n_{Target}, allow\_cycles)$
    end for
    $dsm_R \leftarrow dsm_R \cup column_R$
  end for
  return dsm_R

Figure 8. Pseudocode of the adapted CPM from Clarkson et al. (2004) implemented in the assistance system

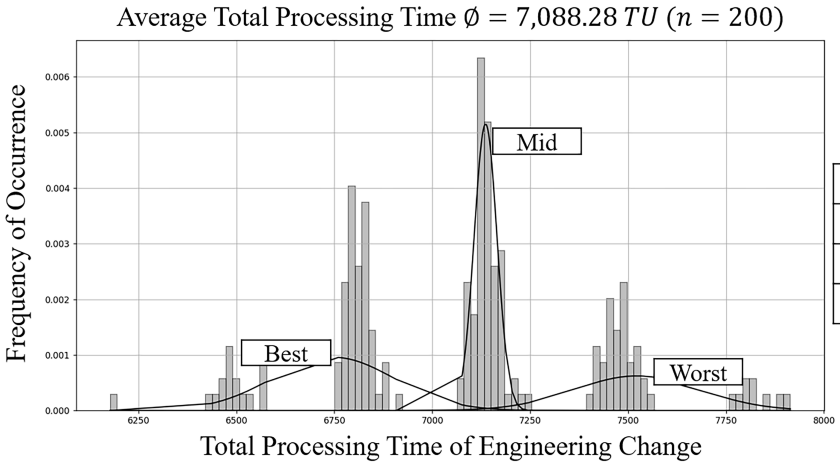
Based on the propagation tree generated in the CPM and the likelihood and impact values from the DSM, the assistance system provides a rough estimation of the time and costs for changing the selected artifact. Figure 7 shows an effort estimation for modifying the artifact ["4_NeoPixel Ring -24x"]. A Monte Carlo simulation provides the average processing time and cost. Additionally, the results from the simulation allow differentiating between a best, mid, and worst-case scenario. The user receives an overview of a scenario's average processing time, cost, and probability.

Figure 9 shows the pseudocode for calculating the time required for implementing a change. For this purpose, the algorithm executes 200 simulation runs for each artifact. It starts with the root of the propagation tree as the first parent. A random value between zero and one is generated for each child of a parent. If the random value is smaller than the likelihood that a change from a parent will propagate to the child, then the processing time for changing the child is accumulated in the variable "total processing time." The processing time is the result of multiplying the initial duration for generating the respective artifact (variable "D" in the algorithm) with the change impact of the parent on the child. For each artifact, the algorithm generates 200 different total processing times. Figure 10 shows the resulting histogram of the total processing time for the engineering change of the artifact ["4_NeoPixel Ring -24x"]. The three scenarios are extracted from the histogram by calculating a Gaussian mixture model with three components using the Python package SciPy (Virtanen et al., 2020) (see Figure 10). All cost values are computed by multiplying the overall average or scenario-specific average total processing time with a predefined cost factor. The weighted output by the algorithm represents the probability of occurrence of the respective scenario concerning the total.

Algorithm 4 Predict Artifact Change Time

Require: $dsm_{PropTree}, dsm^L, dsm^I, D, parent, totalProcessingTime$
  for all $child \in childrenOfParent(dsm_{PropTree}, parent)$ **do**
    $randomValue \leftarrow GenerateRandomValueBetweenZeroAndOne()$
    if $randomValue < dsm^L_{parent,child}$ **then**
      $processingTime \leftarrow D_{parent,child} \cdot dsm^I_{parent,child}$
      $totalProcessingTime \leftarrow totalProcessingTime + processingTime$
      $PredictArtifactChangeTime(dsm_{PropTree}, dsm^L, dsm^I, D, child, totalProcessingTime)$
    end if
  end for
  return $totalProcessingTime$

Figure 9. Pseudocode for predicting the total processing time of a change for one artifact in a single simulation run



Total Cost

$= 7{,}088\,TU \cdot 10\,\frac{€}{TU} = 70{,}882.80€$

Scenario	Time Units	Cost [€]	Weight
Best	6,747.8	67,478	37.2
Mid	7,136.0	71,360	37.7
Worst	7,522.4	75,224	25.1

Figure 10. Interpretation of Monte Carlo Simulation results for rough change time and cost estimation

3.6 Graph Navigation

Due to many artifacts and relationships, the representation generated by the ForceAtlas2 algorithm quickly becomes complex. This can already be seen with a change of the component ["3_SL-Base_printed_large"] in Figure 5. Therefore, the assistance system provides a view allowing users to navigate the path of affected changes. In Figure 11 (left), the user has initiated the view starting with the artifact ["4_NeoPixel Ring -24x"]. The view shows all artifacts directly linked to ["4_NeoPixel Ring -24x"]. By hovering with the mouse over an artifact, the user can see the name of the neighboring artifact. The camera flies over the line by clicking on it, representing the relationship towards the neighboring artifact. From there, the assistance system provides the user with the neighbors of that artifact. This view reduces the graph layout complexity and helps users navigate the graph more efficiently. Through the colors, the user can navigate to those artifacts that are most affected by a change. This allows identifying artifacts possibly influenced by an engineering change.

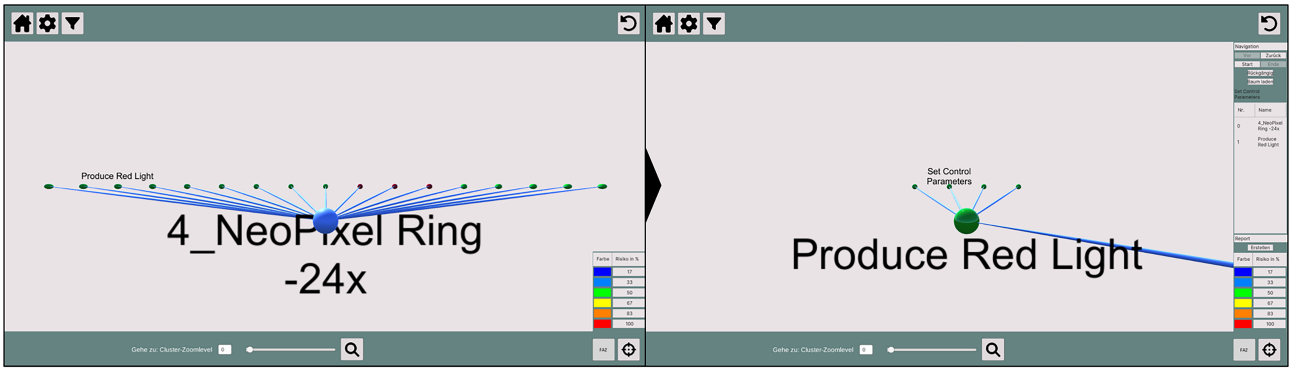


Figure 11. User-guided navigation along the change impact for ["4_NeoPixel Ring -24x"]

4 Discussion and Conclusion

The presented assistance system provides a visual representation and analysis of artifacts and their relationships to each other. The functions of the FuPEP assistance system support the user in identifying and understanding the multiple relationships and possible consequences of a product or process change. When initiating an engineering change, product developers can list and prioritize affected artifacts efficiently. However, existing product development and PLM software tools are superior in organizing and representing artifact information. Therefore, the FuPEP assistance system creates capabilities for visualizing change processes beyond this and represents a valuable extension of these third-party systems.

Visualizing relationships between artifacts, cluster analysis, and change prediction supports product and process developers in managing the effects of artifact changes. The knowledge of relationships between artifacts derived from the visualization is intended to mitigate problems such as outdated documentation, lack of coordination with affected disciplines, delayed changes, etc. (Herrmann et al., 2021). Thus, cluster analyses integrated into the assistance system illustrate the effects of highly modular, integral, or mixed product and process architectures. Furthermore, determining the probability of a change and its impact contributes to estimating the required monetary and temporal effort. This information can be used in many different areas of a development process, like providing a customer with a quick cost estimation of a technical change request in the sales department.

The assistance system is currently still under development. This means that the identified requirements need to be implemented in greater detail. This also concerns an evaluation of the usability of the graphical user interface. So far, only a few lead users of the involved companies tested the usability and functions. The envisaged further development of the assistance system concerns more intuitive guided navigation along the effects of a change. Future activities might involve testing the assistance system's viability throughout the product lifecycle. This includes service and maintenance processes after delivery as well as artifacts and processes from outside the company. Such activities reveal how well the assistance system's functionalities generalize to managing the complexity of product-service systems and supply chains.

Acknowledgments

This research and development project is funded by the German Federal Ministry of Education and Research (BMBF) within the "The Future of Value Creation—Research on Production, Services and Work" program and managed by the Project Management Agency Karlsruhe (PTKA). The authors are responsible for the content of this publication.

Special thanks go to the experts from the German companies Leichtwerk AG and MSF-Vathauer Antriebstechnik GmbH & Co KG for providing valuable feedback in the development of different forms of visualization and for participating in the usability assessments.

References

Bastian, M., Heymann, S., Jacomy, M., 2009. Gephi: An Open Source Software for Exploring and Manipulating Networks, in: Proceedings of the International AAAI Conference on Web and Social Media 3. Nr. 1, pp. 361–362.

Batagelj, V., Mrvar, A., 2004. Pajek—Analysis and Visualization of Large Networks, in: Graph Drawing Software. Springer, Berlin/Heidelberg, Germany, pp. 77–103.

Borjesson, F., Hölttä-Otto, K., 2013. Improved Clustering Algorithm for Design Structure Matrix, ASME 2012 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. pp. 921–930.

Browning, T.R., 2016. Design Structure Matrix Extensions and Innovations: A Survey and New Opportunities. IEEE Transactions on Engineering Management 63(1), 27–52.

Clarkson, P.J., Simons, C., Eckert, C., 2004. Predicting Change Propagation in Complex Design. Journal of Mechanical Design 126(5), 788–797.

Cui, M., 2020. Introduction to the K-Means Clustering Algorithm Based on the Elbow Method. Accounting, Auditing and Finance 1(1), 5–8.

Eppinger, S.D., Browning, T.R., 2012. Design structure matrix methods and applications. MIT press.

Ghoniem, M., Fekete, J.D., Castagliola, P., 2005. On the readability of graphs using node-link and matrix-based representations: a controlled experiment and statistical analysis. Information visualization, 4(2), 114-135.

gRPC. gRPC A high performance, open source universal RPC framework. URL https://grpc.io/

Hagberg, A., Swart, P., Schult, D., 2008. Exploring network structure, dynamics, and function using networkx. (Los Alamos National Lab. (LANL), Los Alamos, NM (United States) LA-UR-08-05495; LA-UR-08-5495)

Herbst, S., Hoffmann, A., 2018. Product Lifecycle Management (PLM) mit Siemens Teamcenter: Grundlagen, Anwendung und Best Practices: Grundlagen, Anwendung und Best Practices. Carl Hanser Verlag GmbH & Co. KG.

Herman, I., Melancon, G., Marshall, M.S., 2000. Graph visualization and navigation in information visualization: A survey. IEEE Transactions on Visualization and Computer Graphics 6(1), 24–43.

Herrmann, J.-P., Imort, S., Trojanowski, C., Deuter, A., 2021. Requirements Elicitation for an Assistance System for Complexity Management in Product Development of SMEs during COVID-19: A Case Study. Computers 10(11), 149.

Idicula, J., 1995. Planning for concurrent engineering.

Imort, S., Pankrath, C., Herrmann, J.-P., Deuter, A., 2022. Verwaltungsschale: Integration einer Design Structure Matrix (DSM) in die VWS. atp Magazin 63, 11-12.
Jacobs, G., Konrad, C., Berroth, J., Zerwas, T., Höpfner, G., Spütz, K., 2022. Function-Oriented Model-Based Product Development. Design Methodology for Future Products, Springer, Cham, 243–263.
Jarratt, T., Keller, R., Nair, S., Eckert, C., Clarkson, P.J., 2004. Visualization Techniques for Product Change and Product Modelling in Complex Design. Springer, Berlin, Heidelberg, 388–391.
Keller, R., Eckert, C.M., Clarkson, P.J., 2006. Matrices or Node-Link Diagrams: Which Visual Representation is Better for Visualising Connectivity Models? Information Visualization 5(1), 62–76.
Latos, B.A., Harlacher, M., Burgert, F., Nitsch, V., Przybysz, P., Mütze-Niewöhner, S., 2018. Complexity Drivers in Digitalized Work Systems: Implications for Cooperative Forms of Work. Advances in Science, Technology and Engineering Systems Journal 3(5), 171–185.
Lloyd, S., 1982. Least squares quantization in PCM. IEEE Transactions on Information Theory 28(2), 129–137.
Macqueen, J., 1967. Classification and analysis of multivariate observations: University of California Press.
Okoe, M., Jianu, R., Kobourov, S., 2018. Node-link or adjacency matrices: Old question, new insights. IEEE transactions on visualization and computer graphics, 25(10), 2940-2952.
Page, L., Brin, S., Motwani, R., Winograd, T., 1999. The PageRank Citation Ranking: Bringing Order to the Web.
Peterson, T., 2015. Understanding Systems through Graph Theory and Dynamic Visualization.
Plattform Industrie 4.0, 2023. Details of the Asset Administration Shell: Specification Part 1–The Exchange of Information between Partners in the Value Chain of Industrie 4.0 (Version 3.0RC02).
Purchase, H., 1997. Which aesthetic has the greatest effect on human understanding? Springer, Berlin, Heidelberg, 248–261.
Scikit-learn, 2011. Machine learning in Python.
Sharman, D.M., Yassine, A.A., 2004. Characterizing complex product architectures. Syst. Eng. (7), 35–60.
Sinha, K., DeWeck, O.L., 2013. A network-based structural complexity metric for engineered complex systems, in: 2013 IEEE International Systems Conference (SysCon), IEEE.
Unity Technologies, 2023. https://unity.com/ (accessed 20.01.2023)
Vessey, I., 1991. Cognitive fit: A theory-based analysis of the graphs versus tables literature. Decision sciences, 22(2), 219-240.
Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., Van der Walt, S.J., Brett, M., Wilson, J., Millman, K.J., Mayorov, N., Nelson, A.R.J., Jones, E., Kern, R., Larson, E., Carey, C.J., Polat, I., Feng, Y., Moore, E.W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E.A., Harris, C.R., Archibald, A.M., Ribeiro, A.H., Pedregosa, F., Van Mulbregt, P., 2020. SciPy 1.0: fundamental algorithms for scientific computing in Python. Nature methods, 17(3), 261-272.
Vogel, W., Lasch, R., 2016. Complexity drivers in manufacturing companies: a literature review. Logistics Research 9(1), 1–66.
Yu, T.-L., Yassine, A.A., Goldberg, D.E., 2007. An information theoretic method for developing modular architectures using genetic algorithms. Research in Engineering Design 18(2), 91–109.

Contact: Jan-Phillip Herrmann, OWL University of Applied Sciences and Arts, Department of Production and Wood Technologies, Campusallee 12, 32657, Lemgo, Germany, +49 5261 702-5474, jan-phillip.herrmann@th-owl.de

Jan-Phillip Herrmann is a research assistant at OWL University of Applied Sciences and Arts and a PhD student at RWTH Aachen University. His research deals with developing user-friendly assistance systems for manufacturing. His PhD thesis is about predicting human decision-making on the shop floor.

Carolin Pankrath is a PLM consultant and senior software engineer at LMtec Service GmbH. She is involved with consulting SMEs on integrating the PLM software Teamcenter, full stack development of Teamcenter extensions, and app development using low-code platforms like Mendix.

Sven Tackenberg is a full professor at OWL University of Applied Sciences and Arts. He holds a PhD in person-centered simulation and optimization of knowledge-intensive services. He supervises several research projects on solutions for product development and manufacturing departments.

Sebastian Imort is a research assistant at OWL University of Applied Sciences and Arts. His main activities comprise developing assistance systems for product development in SMEs and the integration of the Industry 4.0 Asset Administration Shell into business processes for realizing Digital Twins.

Christoph Trojanowski is a research assistant at HTW University of Applied Sciences. His primary interests are in the field of realizing user-friendly interfaces for visualizing dependencies of complex systems in the context of Product Lifecycle Management and Systems Engineering.

Andreas Deuter is a full professor at OWL University of Applied Sciences and Arts. He holds a PhD in designing key figure-oriented software development processes in manufacturing companies. He supervises several research projects on solutions for systems and software engineering.