computers

MDPI

Article

Requirements Elicitation for an Assistance System for Complexity Management in Product Development of SMEs during COVID-19: A Case Study

Jan-Phillip Herrmann [1,*], Sebastian Imort [1], Christoph Trojanowski [2] and Andreas Deuter [1]

[1] Department of Production Engineering and Wood Technologies, OWL University of Applied Sciences and Arts, 32657 Lemgo, Germany; Sebastian.Imort@th-owl.de (S.I.); Andreas.Deuter@th-owl.de (A.D.)
[2] Department of Informatics, Communication, and Economics, University of Applied Sciences, 10318 Berlin, Germany; Christoph.Trojanowski@HTW-Berlin.de
* Correspondence: Jan-Phillip.Herrmann@th-owl.de; Tel.: +49-5261-702-5474

Abstract: Technological progress, upcoming cyber-physical systems, and limited resources confront small and medium-sized enterprises (SMEs) with the challenge of complexity management in product development projects spanning over the entire product lifecycle. SMEs require a solution for documenting and analyzing the functional relationships between multiple domains such as products, software, and processes. The German research project FuPEP "Funktionsorientiertes Komplexitätsmanagement in allen Phasen der Produktentstehung" aims to address this issue by developing an assistance system that supports product developers by visualizing functional relationships. This paper presents the methodology and results of the assistance system's requirements elicitation with two SMEs. Conducting the elicitation during a global pandemic, we discuss its application using specific techniques in light of COVID-19. We model problems and their effects regarding complexity management in product development in a system dynamics model. The most important requirements and use cases elicited are presented, and the requirements elicitation methodology and results are discussed. Additionally, we present a multilayer software architecture design of the assistance system. Our case study suggests a relationship between fear of a missing project focus among project participants and the restriction of requirements elicitation techniques to those possible via web conferencing tools.

Keywords: complexity management; assistance system; product development; systems engineering; design structure matrix; asset administration shell

Citation: Herrmann, J.-P.; Imort, S.; Trojanowski, C.; Deuter, A. Requirements Elicitation for an Assistance System for Complexity Management in Product Development of SMEs during COVID-19: A Case Study. *Computers* **2021**, *10*, 149. https://doi.org/10.3390/computers10110149

Academic Editor: Paolo Bellavista

Received: 11 October 2021
Accepted: 1 November 2021
Published: 10 November 2021

1. Introduction

Product variety, technological change, and process agility are a few of the many complexity drivers in product development and work systems, having substantial effects on time, cost, quality, and flexibility of development projects [1,2]. Adopting early definitions of complexity from a systems theoretic perspective [3,4], a high number of elements and relationships within domains such as products, software, processes, and the organization of manufacturing enterprises varying dynamically over time constitute a complex system. With such systems involved in product development projects and manufacturing processes, small and medium-sized enterprises (SMEs) fail to oversee all complex relationships, leading to missing reconciliation between decision makers and uninformed decision making. Due to engineering changes and iterations, product development misses time and cost objectives [5]. Knowledge about dependencies and propagation of engineering changes through the system resides in domain experts' mental models. Enterprises do not explicitly represent product and software functions in their enterprise information systems, with software being one item in a bill of materials at most. However, functions are the starting point for customer-oriented development, modular system development, and

concretizing the solution space of new product developments [6,7]. The upcoming trend of cyber-physical systems integrating intelligent software functions into physical products adds to the abovementioned challenges [8,9].

Design Structure Matrices (DSMs) represent a promising approach for system decomposition, analyzing product architecture and change propagation through systems, and modeling relationships between multiple domains. However, no existing software toolset allows user-friendly and versatile modeling and analysis of DSMs [10]. From a managerial perspective, Systems Engineering and Product Lifecycle Management (PLM) are two disciplines trying to tackle the challenge of complexity by guiding the development process and managing products over their entire lifecycle [11,12]. A key consideration of these disciplines is engineering change [13], an alteration made to drawings, parts, or software released in the product design process [14]. Additionally, here, current methodologies such as process- and organization-based approaches or computer-aided engineering systems for supporting these disciplines in developing cyber-physical systems reach their limits [8,9].

This paper presents the methodology and results of a requirements elicitation and software design process for developing an assistance system that supports complexity management in SMEs. Such assistance system shall provide a user-friendly and intuitive graphical user interface for visualizing the elements and relationships of complex systems over their entire lifecycle and implement different functions for analysis.

Requirements engineering is a success factor in software projects [15], and the quality of software requirements is strongly related to projects' success [16]. Several methods for requirements elicitation exist, such as ethnography, which involves actively participating in the activities to be optimized, or observation, which involves observing the execution of processes [17]. During the COVID-19 global pandemic, social distancing rules and obligations to work from home restricted the repertoire of requirement elicitation techniques to those that can be executed via web conferences in virtual teams. Employing requirements elicitation techniques requires knowledge sharing in virtual teams, which may be vulnerable to communication breakdowns, mistrust, or conflicts [18]. Therefore, this paper presents a methodology for requirements elicitation that can be applied entirely remotely and discusses its advantages and drawbacks. The paper addresses three subgoals:

1. To describe and discuss the methodology and results of a requirements elicitation for an assistance system for complexity management with two SMEs during the COVID-19 global pandemic;
2. To model the cause-and-effect relationships of complexity management problems in the product development activities of SMEs;
3. To present the assistance system's software architecture using the Asset Administration Shell as a data basis.

The assistance system acts as a supporting tool for SMEs that manufacture complex products and services. Its applicability does not depend on the industrial sector but strongly focuses on companies with an active research and development department. The assistance system is developed in the context of the research project FuPEP "Funktionsorientiertes Komplexitätsmanagement in allen Phasen der Produktentstehung".

In Section 2, an overview of related work in requirements elicitation under COVID-19, existing assistance systems for product development, and visualization of complex systems is given. Gaps in complexity management capabilities and functionalities of existing approaches are identified, and our assistance system's necessity is outlined. The requirements elicitation methodology and results are described in Section 3. A system dynamics model is presented to illustrate cause-and-effect relationships of SMEs' product development problems related to complexity management. The most important requirements and the aggregation of all requirements in several use cases are described. In Section 4, the software architecture is presented based on elicited requirements and use cases, and it is explained how the assistance system uses the Asset Administration Shell as a data management system. Results of Sections 3 and 4 are discussed in Section 5. A conclusion and outlook for future works close the paper in Section 6.

2. Related Work

The literature discusses requirements elicitation without face-to-face communication between participants in distributed and global requirements elicitation, sometimes referred to as virtual teams. The authors in [19] suggest a set of strategies to deal with communication problems in global requirements elicitation related to cultural differences, means of communication such as ontologies, and technology selection. The authors in [20] evaluate the effectiveness of different requirements elicitation techniques in a distributed setting using groupware (software for collaboration), with the question-and-answer method and use cases as the two most popular. More generally, several studies discuss problems of knowledge sharing, collaboration challenges, and strategies for mitigation in virtual teams [18,21], with more recent studies considering the challenges of virtual teams in the context of a global pandemic [22]. Few studies were identified dealing with requirements elicitation under the COVID-19 global pandemic directly. Closest to our contribution come [23], reviewing cost-effective requirements elicitation techniques during COVID-19 where requirements engineers are unable to interact with the customer of the final system. The authors conclude with introspection as the most cost-effective method, followed by surveys, brainstorming, interviews, and joint application development. The authors identified no further previous works about requirements elicitation during COVID-19. In particular, there was a lack of works trying to apply and report the success of requirements elicitation techniques under these circumstances. This leads us to present a case study in which we perform all requirements elicitation activities via web conferencing tools with two SMEs.

Previous works that present actual functionalities of assistance systems for product development deal with specific optimization activities during the development process. One data-driven assistance system aims at closing the gap between expected and actual product properties in new product development. The assistance system helps product developers to reapply expert knowledge and solutions to meet the desired properties by analyzing the historical data of past product generations using machine learning methods [24,25]. A self-learning and knowledge-based assistance system described in [26] supports product developers in product design for the manufacturing process of sheet-metal parts in the early phases of product development. The assistance system offers design features for a sheet-metal part to the design engineers in a synthesis step. In an analysis step, it evaluates the sheet-metal part consisting of the design features, quantitatively considering a set of target values. The assistance system incorporates data mining methods for knowledge discovery using data stored in its database to enable the analysis. Explicit support for engineering change management and different functionalities for helping product developers provides the PLM software Teamcenter developed by Siemens [27]. Offered capabilities are, e.g., managing designs, managing revisions, and establishing relationships between different artifacts. However, this software does not aim at intuitively visualizing and analyzing change propagations of complex systems as a complexity management support for product developers. Considering the assistance systems in product development reviewed above implies the same gap of such missing functionality.

Studies that investigate visualizing complex systems build on DSMs and representations from graph theory. In [28], a link connection plot (also referred to as molecular diagram elsewhere [29]) illustrates the system elements as labeled nodes and their relationships as directed edges. A component connection plot shows a specific component of a system at its center surrounded by all components with a direct relationship. Using the same graphical visualization technique, [30] present the capabilities of DSMs and network analysis to visualize complex systems as organized and clustered graphs. Freely available tools such as Gephi (open source) [31] or Pajek [32] for non-commercial use exist, providing functionalities for network analysis and visualization. To the best of the authors' knowledge, no assistance system for complexity management exists, providing an intuitive and user-friendly visualization of complex systems, integrating network and

change propagation analyses, and management functionalities for its implementation in an organizational context.

3. Requirements Elicitation in SMEs for a Complexity Management Assistance System

3.1. Methodology

The requirements elicitation was carried out from 18 January 2021 until 5 May 2021 with two German medium-sized enterprises by conducting semi-structured interviews via the web conference tool Cisco Webex. Company A is involved with the development and prototyping of complex systems in the aerospace industry. Company B is a manufacturer of complex systems in the drive technology and automation industry.

In twelve interviews, eleven employees of Company A from the product development, manufacturing, and quality assurance department were interviewed individually (one employee was interviewed twice). In two group interviews, five employees from the product development and manufacturing department and the managing director of Company B were interviewed. Next to the employees of Company A and Company B, two PLM experts from two additional companies offering components, systems, and software solutions for electronics, automation and industrial digitalization were interviewed.

The employees and experts were asked for concrete problems arising due to engineering changes and their effects on other products, departments, or processes and their requirements for the assistance system. An interview guide was developed oriented at recommendations and guidelines from the literature [33–35], with seven open questions and additional follow-up questions supporting the interviews. Following the recommendations of [36], a pilot test of the interview guide was conducted with two professors of the Ostwestfalen-Lippe University of Applied Sciences and Arts for its adaptation and optimization. The interview guide starts by introducing the interviewee to the topic and interview purpose as well as a warm-up question about the interviewee's function in the enterprise. The first open question asks for concrete situations in day-to-day business, in which engineering changes had significant effects, e.g., in the form of errors in other departments or activities. This question shall motivate the interviewee to give detailed verbal insights to past events as a basis for further questions and adopts the ideas of the Critical Incident Technique and Grand-Tour Question [34,37].

One person conducted the interview by asking questions using the developed interview guide. A second person protocolled the interview, and a third person listened carefully for optionally asking follow-up questions. Additionally, all interviews were recorded using the recording functionality provided by Cisco Webex to capture all relevant information mentioned. Two persons documented problems, their causes, needs, requirements, and potential use cases of the assistance system, while listening to each interview one more time individually and independently, ensuring a four-ears-principle. For requirements documentation, Rupp's formulation template [38], extended by the events and states of a requirement according to the Easy approach to requirements syntax [39], was used. Afterward, the two persons merged their documented requirements in a requirements stack. In a web conference meeting, the requirements stack (a word document with empty tables for each requirement, including meta-information such as requirement ID, author, date of creation) was shared using screen sharing. The two persons presented their requirements in alternating order. For each requirement readout, it was checked if the requirement was already part of the requirements stack. If the readout requirement was not a new one or did not contribute to any requirement of the requirements stack, it was neglected. Otherwise, it was added to the requirements stack, or the new content extended an existing requirement. Merging the requirements stack in that manner was perceived to be manageable and efficient for the two persons up to a number of 100 requirements. However, the more this number was exceeded, the more checking if a requirement already exists required significant effort. The formulation of each requirement was conducted considering a checklist of quality characteristics for requirement formulation [40].

After requirements formulation, as proposed in [41], a revision for identifying and eliminating inconsistencies, conflicts, and incompleteness in the requirements was conducted for requirements validation. Due to the high number of requirements, use cases each aggregating a set of requirements were formulated. In two separate web conference meetings, these use cases were presented, discussed, and revised with the interviewees from Company A and Company B.

Next, each use case was prioritized using a standardized table. Each interviewee from Company A and Company B was asked via e-mail to assess the priority and severity of each use case in the standardized table on a three-level ordinal scale. The higher the priority (levels: "low", "medium", "high") of a use case, the earlier it was implemented into the assistance system in the implementation phase. Three levels of severity were defined: (1) "nice to have", wishes that should not necessarily be implemented but would contribute to end-user enthusiasm; (2) "should have", wishes, which can be implemented; and (3) "must have", required, must be implemented into the assistance system. After receiving the use case prioritization from the interviewees, requirements elicitation was completed.

3.2. Results

3.2.1. System Dynamics Modeling of Complexity Management Problems in SMEs

Before describing the requirements and use cases, we present problems related to complexity management in the product development of SMEs. Next to eliciting requirements from the interviews with Company A and Company B, the cause-and-effect relationships of these problems could be retrieved. We use the concept of a system dynamics model (SDM) developed by Jay Wright Forrester [42]. SDMs help to model the structure and behavior of systems, in particular, modeling socioeconomic systems for supporting management decisions. An SDM is a causal loop diagram for the qualitative representation of cause-and-effect relationships. SDMs are generally auto- and cross-correlated and contain loops and delays as relationships between the systems' components. Figure 1 shows the SDM elicited through interviews with Company A and Company B. The SDM contains positive relationships only and no loops. "Relationships unknown" between system elements represents a major source and sink for variables, and "deviations between implementation and requirements" a major sink. Figure 2 illustrates the first and second-order effects of unknown relationships on time, cost, and quality variables.

3.2.2. Requirements Overview

From the interviews, 130 requirements could be retrieved. Because of the high number of requirements, the most important ones are presented here. On the one hand, requirements are considered important when they were mentioned by at least four interviewees, called multiply mentioned requirements in the further course, and on the other hand, requirements that are part of high priority and high severity use cases. Requirements with multiple mentions are listed by name in Table 1.

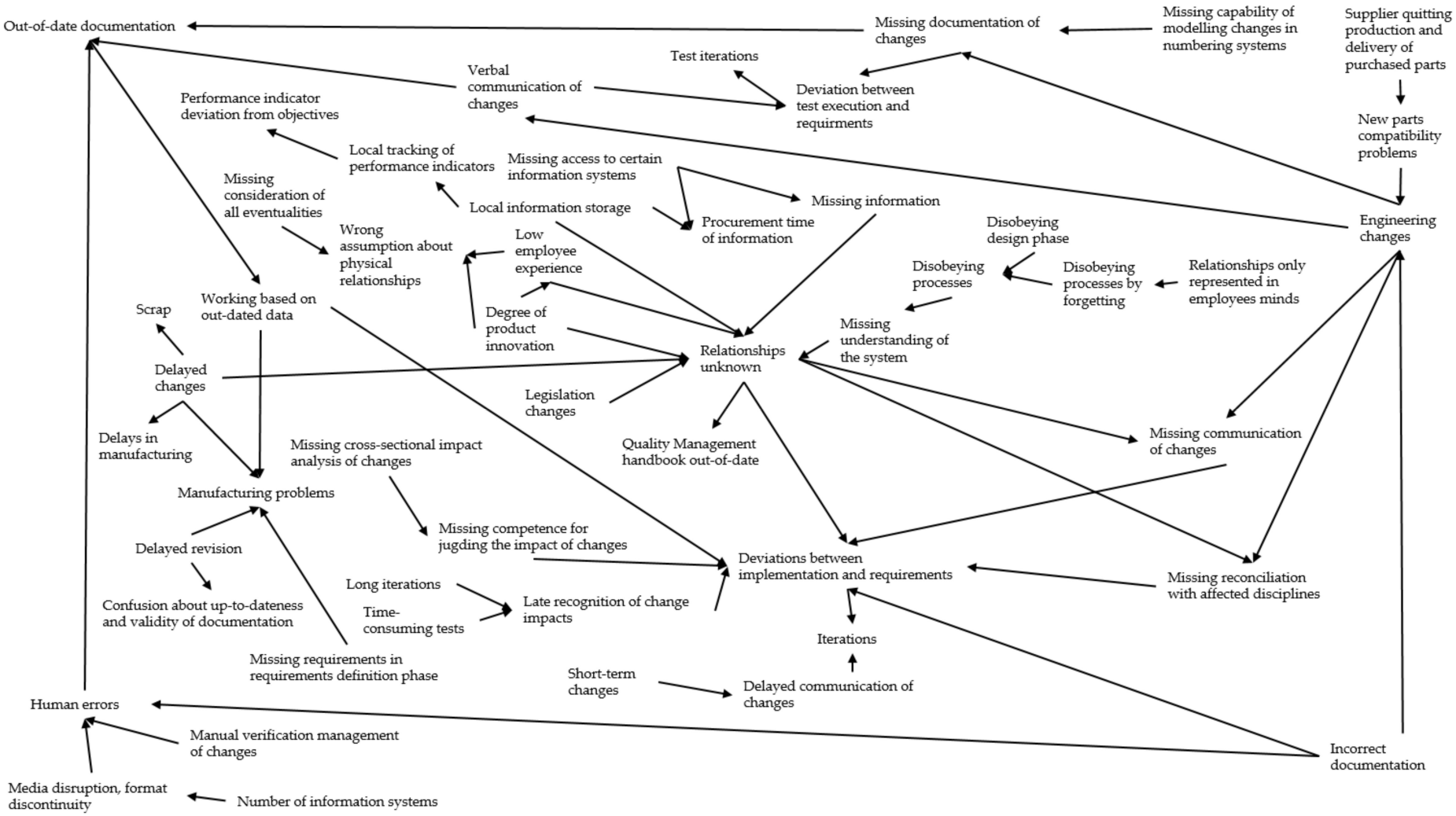

Figure 1. SDM of problems related to complexity management in Company A and Company B.

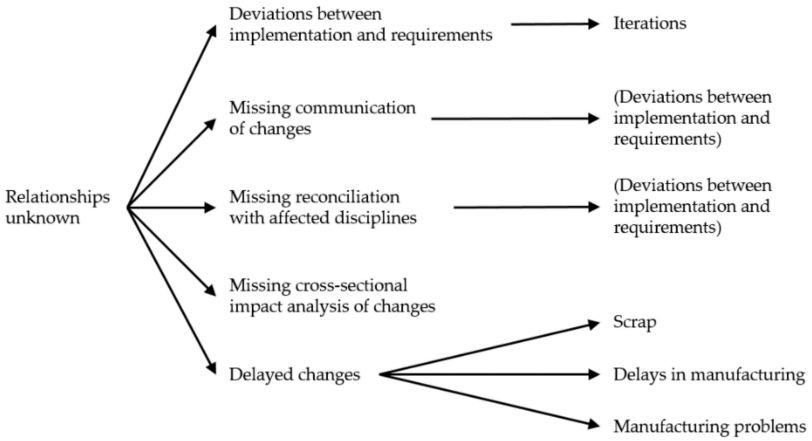

Figure 2. Effects of unknown relationships on variables related to time, cost, and quality.

Table 1. Overview of multiply mentioned requirements (REQ) with their respective title.

Multiply Mentioned Requirements
REQ 02—Documentation and communication of artifact changes
REQ 05—Representation of relationships between artifacts
REQ 10—Role and department view
REQ 19—Documentation of artifact changes
REQ 23—Display the product life cycle status of a product
REQ 46—No pressure to decide
REQ 55—Support for compliance with standardized product development processes
REQ 60—Communication of artifact changes to affected departments
REQ 81—Communicate change impulse

The requirements can be divided into organizational (REQ 10, REQ 23, REQ 55), artifact-related (REQ 02, REQ 05, REQ 19), where REQ 60 and REQ 81 can be interpreted as child-requirements of REQ 02, and non-functional requirements (REQ 46). Artifact-related requirements refer directly to the artifacts or their change. In the project context, artifacts include all elements that can be subject to changes: requirements, documents, products or components, (software) functions, and processes. The requirements that form the basis of the prioritized use cases can be found in Table 2, associated with the underlying use cases that are examined in more detail in the following section.

Table 2. Requirements assigned to the most important use cases (UC).

Use Case	Requirements
UC001—Making dependencies visible	REQ 005—Representation of relationships between artifacts REQ 006—Representation of relationships between documents and construction files REQ 007—Traceability of defective materials REQ 042—Representation of relationships between documents of the development manual REQ 043—Traceability from requirements elicitation to design REQ 054—Display of legally indispensable components for new developments REQ 061—Display of most recent documents during documentation viewing REQ 064—Display of effects in case of requirement change REQ 071—Comply and display reporting channels REQ 079—Differentiation of requirements in development phases REQ 082—Provide test relevant information REQ 085—Overview of documents associated with the component REQ 092—Material usage statement REQ 096—Relationship between requirements, designs and test cases REQ 100—Point-and-click-action REQ 117—Graphical representation of assemblies REQ 121—Information display in assembly structure REQ 122—Relationships between departments of the product development process REQ 125—Show information when opening artifact
UC008.a—system function give approval	REQ 026—Automatic assignment of document number REQ 039—Digital release of documents REQ 115—Error prevention through release, check and test processes
UC012.a—Find information	REQ 044—Top-down and bottom-up view of relationships between documents
UC012.b—Represent information	REQ 023—Display the product lifecycle status of a product REQ 051—Information display via dashboard REQ 052—Error management via dashboard REQ 068—Categorization of document relationships by subject area in tree structure REQ 073—Display and compare artifact versions REQ 091—Traffic light system at taskbar REQ 094—Change impact analysis checklist REQ 107—Theme classification on dashboard REQ 108—Traceability matrix REQ 121—Information display in assembly structure
UC012.f—Represent changes	REQ 089—Display change warning REQ 091—Traffic light system at taskbar REQ 094—Change impact analysis checklist

3.2.3. Use Cases Overview

Based on the 130 requirements, 30 use cases were created. Each use case aggregates a set of requirements. However, use cases do not necessarily aggregate the same number of requirements, and one requirement can belong to more than one use case. A use case documents its ID, name, purpose, trigger, outcome, and a sequence of activities description. All requirements were clustered according to subject areas and then described in use cases. The use cases formulated are given with their ID, description and actor in Table 3.

Table 3. 30 use cases formulated based on the requirements.

Use Case ID	Description	Actor
UC001	Making dependencies visible	Developer (Product)
UC002.a	Manage user roles	Administrator
UC002.b	Configure rights management	Administrator
UC002.c	Subscribe to artifacts	System user
UC002.d	Configure notifications	System user
UC003.a	Submit changes	System user
UC003.b	(Organizational) evaluation and traceability of a change	System user
UC003.c	(Lifecycle) traceability of change effects	System user
UC003.d	Changes by external source	External users
UC003.e	Limit change options	System user
UC003.f	Show status of change	System user
UC004.a	Communicate change	System user
UC004.b	Communicate change via warning	System user
UC004.c	Communicate change—Task management	System user
UC005	Communicate errors	System user
UC006	Create templates	System user
UC007	Perform tests	Quality assurance
UC008.a	System function—Give approval	System user
UC008.b	System function—Create comments	System user
UC008.c.1	System function—Create plausibility checks	Developer (Software)
UC008.c.2	System function—Carry out plausibility checks	System user
UC009	System function—Make or carry out decision proposals	System user
UC010	System function—Obtain external system data	Third-party system
UC011	System function—Start third-party systems	Third-party system
UC012.a	Find information	System user
UC012.b	Represent information	System user
UC012.c	Filter information	System user
UC012.d	Show notifications	System user
UC012.e	Display artifact status	System user
UC012.f	Represent changes	System user

The use cases describe possible scenarios in which the assistance system can support employees in managing complexity in their companies. The actors comprise those determined by the interviewees and those determined based on the third-party systems used in the companies or their external partners. The actor System User comprises the roles: Engineer, Administrator, Quality Assurance, and Developer. The roles Engineer and Developer further subdivide into enterprise-specific actors such as Systems Engineer or Materials Manager, which we consider too specific for including them in the overview.

The prioritization of all use cases as the average interviewee assessment illustrates Figure 3. As can be observed, use cases with a low (high) priority were also evaluated with a low (high) severity. The prioritization serves as a basis for deciding which use cases and respective requirements will be implemented first in the implementation phase of the assistance system development. In the given prioritization, use cases UC001, UC008.a, UC012.a, UC012.b, UC012.f and UC003.b represent those with the highest priority and severity.

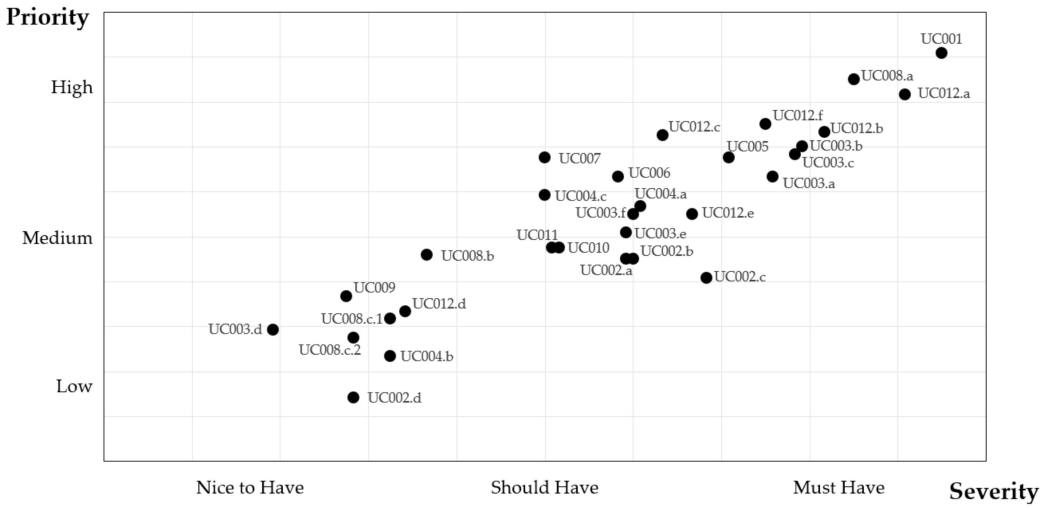

Figure 3. Use case prioritization regarding priority and severity.

UC001 aims to make relationships between elements of complex systems and across domains visible in the assistance system. The assistance system collects all relevant data regarding requirements, processes, products, documents, functions, organization, designs, test cases, construction files, and documents of the development manual, departments, and components and establishes relationships between them. If a generation of relationships automatically through the assistance system is not possible, assistance system users can define relationships. UC008.a aims to provide functionalities for document approval to the users. Use Cases UC012.a, UC012.b, UC012.f, and UC003.b aim at supporting the information search on artifacts, enable the display of the information and provide functionalities of propagation analyses of engineering changes. Thus, it must be possible for the user to search specifically for artifacts or their characteristics and parameters in the assistance system. The user must be able to select and adapt the appropriate view for the information found. If the user wants to initiate a change to the information found, the assistance system must be able to evaluate the effects on other artifacts in terms of time, costs, severity, and probability and present it in an appropriate form. The presentation can be on an ordinal scale for coarse evaluation, in a table for finer evaluation, or in a three-dimensional space for evaluation with a depiction of dependencies.

4. Assistance System for Complexity Management in SME Architecture

4.1. Asset Administration Shell for Data Management

The assistance system aims to provide support for small and medium-sized enterprises. From experience, companies from this category often use software from different providers for diverse tasks. Due to this internal IT structuring, the accruing data are often only accessible in different formats. For the assistance system, a uniform data model must, therefore, be used to standardize the information of a company's system landscape.

For this reason, the data model of the Asset Administration Shell (AAS) was agreed upon for the assistance system's data management. The German consortium "Plattform Industrie 4.0" was the first to present the concept of the AAS under the name Digital Twin [43]. Based on the resulting findings, the Industrial Digital Twin Association (IDTA) [44] was founded.

The AAS is to be understood as the digital representation of an asset, whereby all data from the product life cycle can be taken into account and find a place in the data model. In this definition, everything having value for the company can be understood as an asset, regardless of being physical or digital.

The AAS data model consists of three main components (Figure 4): asset class, view class, and the submodel class. The asset class defines which asset the AAS refers to and its type (type or instance). Through the view class, the information of the AAS can be

limited to a certain view to filter information for display. The submodel class contains the actual data of the asset. The submodels consist of submodel elements that the Industry 4.0 platform has already developed. There is no maximum for the number of submodels or the elements they contain. It can therefore be adapted as desired to the amount of data available. To make data available to the assistance system, the submodels are used to store the data of the artifacts such as drawings, functions, and product components, to name a few.

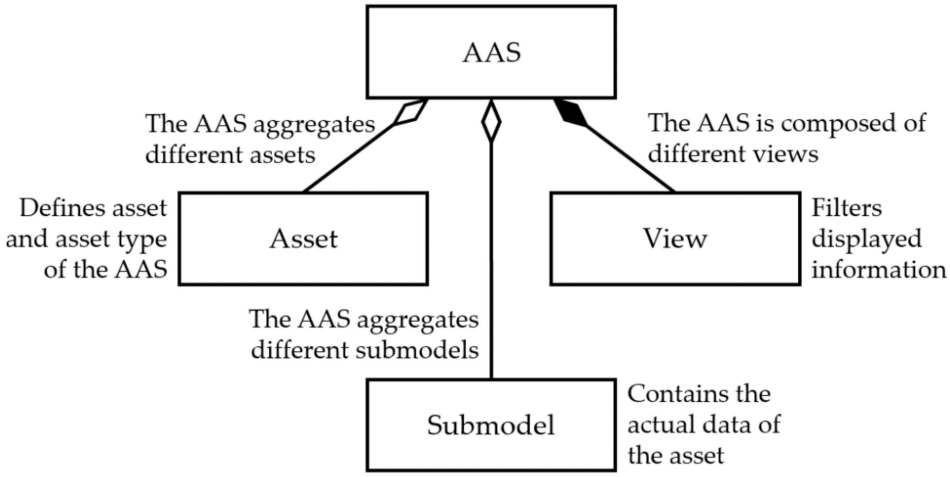

Figure 4. Extract from the AAS Meta Model oriented at [43].

In addition to the classes presented here, the meta-model of the AAS consists of other classes that refer to, e.g., security, types, and submodel elements. However, these classes are of no further relevance in the remainder of this section. Furthermore, there is already documentation on export formats such as XML (Extensible Markup Language) and JSON (JavaScript Object Notation). The Industry 4.0 platform provides a class diagram for the management shell but no direct implementation. Therefore, initiatives for implementations have been realized by research projects (TeDZ, BaSys 4.0) and companies (AASXPackage Explorer, AASX-Server).

The Package Explorer has a graphical user interface that can be used to create and manage an administration shell manually. In addition, the Package Explorer offers the possibility of importing and exporting data from various formats and making these data available via a server. Communication protocols that have already been implemented include OPC UA and Rest.

The structure of the submodels and submodel elements specified by the Industry 4.0 platform enables the user to create individual submodels. In this context, standardization processes are currently under debate. So far, only the submodel "digital type plate" has been implemented in this form [44]. One task in implementing the assistance system is, therefore, the creation of corresponding submodels for transferring the data of the external systems into corresponding internal structures.

4.2. Software Architecture

An intuitive and user-friendly visualization through the assistance system shall be realized with a 3D user interface based on Unity, a cross-platform video game engine for developing two- and three-dimensional games [45]. Next to a 3D interface, the assistance system shall support DSM algorithms for clustering complex systems and change propagation analysis in a DSM logic. Since several interfaces and users potentially have to access functionalities of the DSM logic, the DSM logic is implemented into a separate server. The use of the AAS requires a further component that provides the hosting of this. Since the components listed so far are in a mutual client–server relationship, communication

protocols for sending and receiving information and converters for converting the internal classes into the JSON format must also be considered.

In addition to the components already mentioned, further components can be derived from the requirements and use cases collected in the requirements elicitation:

1. REQ 10/UC002.a—Role and department view
   - The keyword "Role" implies a role and rights management. Therefore, a database or an interface to an external system must exist which can take over this task.
   - Impact: User database or rights database.
2. REQ 60/UC004.a—Communication of artifact changes to affected departments
   - Automated communication that reacts to changes of specific objects implies another component that monitors all existing artifacts and informs users about changes according to their subscriptions.
   - Impact: Communication component, subscription database.
3. Section 3.2.3. Use Case Overview
   - Based on the use cases and the actors mentioned therein, interfaces to external authoring systems are necessary. These have so far been grouped under PLM, ALM, and ERP.
   - Effects: Interface to external systems.

These aspects now result in a more complete picture of the system architecture of the assistance system (Figure 5).

The system architecture in Figure 5 consists of four system blocks. The Unity, DSM server, and AASX server blocks form the three-layer architecture of the actual assistance system. The Unity block provides the 3D interface, that is, the interaction interface between user and system. The DSM server contains the logical processes for evaluating data from the AAS with the help of DSM algorithms. The AASX server is the data consolidation layer, in which all the company's information is converted into a uniform data model. The information carrier in this diagram is the corporate network, which provides both the authoring systems and databases for users and artifacts. The last element in the software architecture is the monitoring system, which constantly monitors the authoring systems and informs the users of any changes.

The diagram also shows that all components are in a client–server relationship. The top level is the client of the level below. Through this arrangement, the information flow of the system could also be shown.

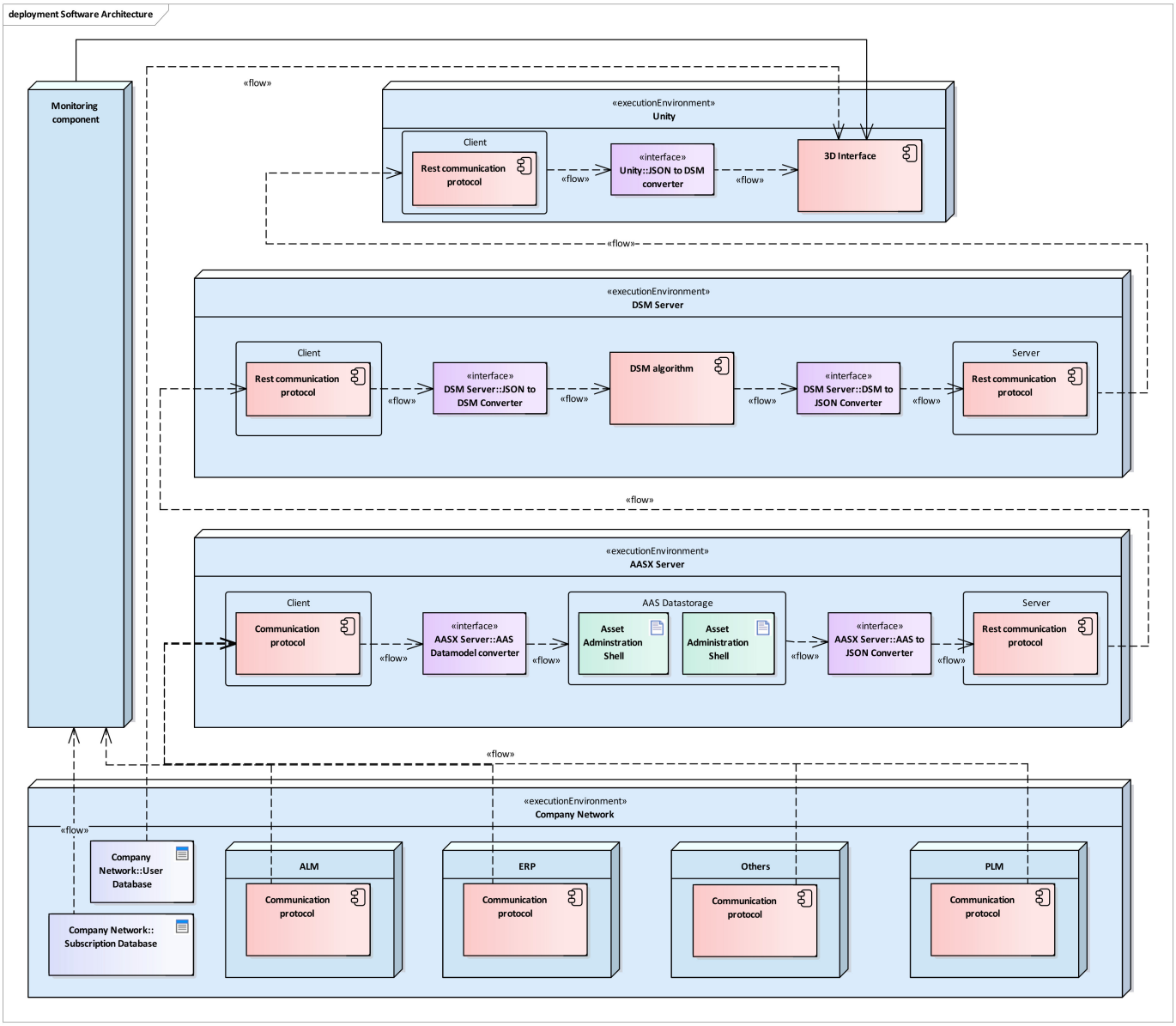

Figure 5. Software Architecture.

5. Discussion

5.1. Implications

This paper presents the elicitation of requirements for an assistance system for complexity management in SMEs under the COVID-19 global pandemic. It describes a methodology of requirements elicitation that can be conducted using only web conferencing tools. The combination of two methodologies for requirements formulation, namely, Rupp's formulation template and the Easy approach to requirements syntax, as well as a requirements stack methodology for merging elicited requirements, are proposed. Practitioners can implement these adaptations in their methodology when facing a global pandemic.

Problems related to complexity management in product development of SMEs and their cause-and-effect relationships are presented in the SDM. The modeled problems mainly refer to organizational issues and human-related challenges in the product development of SMEs. Researchers can investigate how to solve these problems by employing more human-centered methodologies while considering new paradigms such as advanced systems engineering [46]. The presented requirements and prioritized use cases concretize

these future research needs, particularly for developing an intuitive and user-friendly graphical user interface for complexity management.

The multilayer software architecture described in this paper gives first insights into the assistance system's implementation. It contributes to recent research in PLM by providing a potential solution for consolidating data from different third-party systems. The inclusion of the AAS in the architecture implies the definition of DSM-specific submodels. Thus, the architecture provides the basis for future developments of submodels that can be used for complexity management focused on SMEs.

5.2. Limitations

Restricting the available requirements elicitation techniques to those possible during the pandemic represents a strong limitation of the presented methodology. Conducting interviews led to several requirements and use cases. However, fear of a missing focus of the development project arose in the project team. This could be due to missing techniques such as ethnography or observation of product development processes directly at the companies' sites. Possible results of these techniques could have been flaws and weaknesses regarding complexity management identified in modeling development processes, a better understanding of end-users' needs, and a precise definition of the activities in which the assistance system supports end-users. Another reason for fear of a missing focus could be the first eliciting requirements and then formulating use cases based on the requirements. For example, the unified modeling language prescribes conducting these activities vice versa, that is, requirements should be formulated based on elicited high-level use cases [47].

Another limitation is conducting the requirements elicitation only with interviewees from two SMEs. The interviews with PLM experts from two additional companies complemented the results and gave positive evidence of the representativeness of the elicited requirements. However, interviewing SMEs from other industrial sectors developing complex products might give more refined and thorough requirements. Additionally, the different numbers of interviewees for company A and company B might impose a stronger focus of the requirements on problems and needs of only one company.

Conducting the requirements elicitation techniques solely via web conferencing tools did not pose any problem in comparison to applying the same techniques with telephone or face-to-face communication. Negative effects such as communication breakdowns, mistrust, or conflicts but also positive effects such as overcoming time and space limitations are discussed in the literature in the context of web conferencing [18]. From our experience, none of the discussed problems arose, the mood was perceived positively throughout conversations, and it was easy to build rapport.

As mentioned in Section 3.1, merging the requirements written down by two persons while listening to the interview recordings one more time in the requirements stack poses the problem of a high cognitive effort if the number exceeds 100 requirements. It should be investigated how to merge requirements when the requirements stack already contains more than 100 requirements. Additionally, Section 3.1 describes how a quality characteristics checklist for requirement formulation was used. However, the checklist was not used actively during formulation but read once before starting with the formulation. Thus, not all quality characteristics were equally and reliably considered. Here, we stress the need to check requirements against the quality characteristics during requirements formulation properly.

Requirements were prioritized along with the dimensions of priority and severity. Interviewees did not differentiate between these two dimensions considerably. Plotting the prioritization results in a two-dimensional graph could be approximated with a linear function with a positive gradient of one (see Figure 3). Thus, priority and severity were assigned the same ordinal values. Future works could employ more sophisticated requirements prioritization techniques than those used in this paper.

As mentioned in Section 3.2.1, the SDM of complexity-related problems and cause-and-effect relationships in product development of SMEs has not been validated using SDM

validation techniques such as direct-structure tests [48]. The SDM gives a comprehensive overview of these problems and their relationships. However, we stress that it should not be interpreted as a validated and completely accurate model in its presented form.

6. Conclusions

We have presented the requirements elicitation for an assistance system for complexity management in product development of SMEs during COVID-19. We described the applied methodology and gave an overview of complexity management problems and their effects on product development and manufacturing models in an SDM. We listed the most important requirements of the assistance system and showed all use cases formulated based on the requirements. The requirements and use cases reflect the gap of existing assistance systems not providing intuitive and user-friendly visualizations for complex product relationships. Next, we discussed the implications and limitations of the requirements elicitation methodology and results. In our case study, the main effect of COVID-19 on the requirements elicitation methodology is the restriction to techniques that do not involve any activities at the end-users' company site. It suggests that business use cases accurately describing the assistance system's use in product development processes could not be formulated because of the pandemic, leading to fear of a missing focus among the development project's participants. Finally, we presented the software architecture of an assistance system, implementing the requirements and use cases incorporating the AAS for data management.

Future works involve detailing the assistance system's architecture. The assistance system will be implemented based on the requirements and use cases presented in this paper. DSM algorithms for clustering and analyzing complex systems, complexity reduction, and propagation analyses of engineering changes will be formulated. Finally, pilot testing of the assistance system will be performed, accompanied by usability assessments of its employment in daily routine.

Author Contributions: Writing—original draft, methodology, and investigation, J.-P.H.; Writing—original draft, software, investigation, S.I. and C.T.; Project administration and supervision, A.D. All authors have read and agreed to the published version of the manuscript.

Funding: 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.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available in article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Vogel, W.; Lasch, R. Complexity drivers in manufacturing companies: A literature review. *Logist. Res.* **2016**, *9*, 1–66. [CrossRef]
2. Latos, B.A.; Harlacher, M.; Burgert, F.; Nitsch, V.; Przybysz, P.; Niewöhner, S.M. Complexity Drivers in Digitalized Work Systems: Implications for Cooperative Forms of Work. *Adv. Sci. Technol. Eng. Syst. J.* **2018**, *3*, 171–185. [CrossRef]
3. Ulrich, H.; Probst, G.J.B. *Anleitung zum Ganzheitlichen Denken und Handeln: Ein Brevier für Führungskräfte*, 4th ed.; Haupt: Bern, Switzerland, 1995.
4. Von Bertalanffy, L. Vorläufer und begründer der systemtheorie. Berl. Colloq. 1971, 17–28. Available online: https://www.bibsonomy.org/bibtex/28cef65f75bf261584a080dcae459936f/isa (accessed on 2 November 2021).
5. Browning, T.R.; Eppinger, S.D. Modeling impacts of process architecture on cost and schedule risk in product development. *IEEE Trans. Eng. Manag.* **2002**, *49*, 428–442. [CrossRef]
6. Mattmann, I. *Modellintegrierte Produkt-und Prozessentwicklung*; Springer Fachmedien Wiesbaden GmbH: Wiesbaden, Germany, 2017.

7. Renner, I. Methodische Unterstützung Funktionsorientierter Baukastenentwicklung am Beispiel Automobil. Ph.D. Thesis, Technische Universität München, München, Germany, 2007. Available online: https://mediatum.ub.tum.de/doc/627386/file.pdf (accessed on 2 November 2021).
8. Törngren, M.; Sellgren, U. Complexity Challenges in Development of Cyber-Physical Systems. In *Principles of Modeling*; Springer: Cham, Switzerland, 2018; pp. 478–503. [CrossRef]
9. Törngren, M.; Grogan, P. How to Deal with the Complexity of Future Cyber-Physical Systems? *Designs* **2018**, *2*, 40. [CrossRef]
10. Browning, T.R. Design Structure Matrix Extensions and Innovations: A Survey and New Opportunities. *IEEE Trans. Eng. Manag.* **2016**, *63*, 27–52. [CrossRef]
11. Kossiakoff, A.; Sweet, W.N.; Seymour, S.J.; Biemer, S.M. *Systems Engineering Principles and Practice*; John Wiley & Sons: Hoboken, NJ, USA, 2011.
12. Stark, J. Product Lifecycle Management. In *Product Lifecycle Management*, 2nd ed.; Stark, J., Ed.; Springer: Cham, Switzerland, 2016; pp. 1–35. [CrossRef]
13. Jarratt, T.A.W.; Eckert, C.M.; Caldwell, N.H.M.; Clarkson, P.J. Engineering change: An overview and perspective on the literature. *Res. Eng. Des.* **2011**, *22*, 103–124. [CrossRef]
14. Jarratt, T.; Clarkson, J.; Eckert, C. Engineering change. In *Design Process Improvement*; Springer London: London, UK, 2005; pp. 262–285. [CrossRef]
15. Hofmann, H.F.; Lehner, F. Requirements engineering as a success factor in software projects. *IEEE Softw.* **2001**, *18*, 58–66. [CrossRef]
16. Kamata, M.I.; Tamai, T. How Does Requirements Quality Relate to Project Success or Failure? In Proceedings of the IEEE International Requirements Engineering Conference, Delhi, India, 15–19 October 2007; pp. 69–78. [CrossRef]
17. Zowghi, D.; Coulin, C. Requirements Elicitation: A Survey of Techniques, Approaches, and Tools. In *Engineering and Managing Software Requirements*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 19–46. [CrossRef]
18. Rosen, B.; Furst, S.; Blackburn, R. Overcoming barriers to knowledge sharing in virtual teams. *Organ. Dyn.* **2007**, *36*, 259–273. [CrossRef]
19. Aranda, G.N.; Vizcaíno, A.; Cechich, A.; Piattini, M. Strategies to Minimize Problems in Global Requirements Elicitation. *CLEI Electron. J.* **2008**, *11*, 1. Available online: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.566.6607&rep=rep1&type=pdf (accessed on 2 November 2021). [CrossRef]
20. Lloyd, W.J.; Rosson, M.B.; Arthur, J.D. Effectiveness of elicitation techniques in distributed requirements engineering. In Proceedings of the IEEE Joint International Conference on Requirements Engineering, Essen, Germany, 9–13 September 2002; pp. 311–318. [CrossRef]
21. Morrison-Smith, S.; Ruiz, J. Challenges and barriers in virtual teams: A literature review. *SN Appl. Sci.* **2020**, *2*, 1–33. [CrossRef]
22. Feitosa, J.; Salas, E. Today's virtual teams: Adapting lessons learned to the pandemic context. *Organ. Dyn.* **2021**, *50*, 1–4. [CrossRef] [PubMed]
23. ul Amin, T.; Shahzad, B.; Fazal e, A.; Shoaib, M. Economical Requirements Elicitation Techniques During COVID-19: A Systematic Literature Review. *Comput. Mater. Contin.* **2021**, *67*, 2665–2680. [CrossRef]
24. Küstner, C. *Assistenzsystem zur Unterstützung der datengetriebenen Produktentwicklung*; FAU University Press: Erlangen, Germany, 2020. Available online: https://opus4.kobv.de/opus4-fau/files/14597/Kuestner_Diss_MB_353.pdf (accessed on 2 November 2021).
25. Kuestner, C.; Wartzack, S. The realization of an engineering assistance system for the development of noise-reduced rotating machines. In *DS 80-4 Proceedings of the 20th International Conference on Engineering Design (ICED 15), Design for X, Design to X, Milan, Italy, 27–30 July 2015*; Volume 4, pp. 71–80. Available online: https://www.designsociety.org/publication/37771/ (accessed on 2 November 2021).
26. Sauer, C.; Breitsprecher, T.; Küstner, C.; Schleich, B.; Wartzack, S. SLASSY—An Assistance System for Performing Design for Manufacturing in Sheet-Bulk Metal Forming: Architecture and Self-Learning Aspects. *AI* **2021**, *2*, 307–329. [CrossRef]
27. Herbst, S.; Hoffmann, A. *Product Lifecycle Management (PLM) mit Siemens Teamcenter: Grundlagen, Anwendung und Best Practices*; Carl Hanser Verlag GmbH Co KG: München, Germany, 2018.
28. Jarratt, T.; Keller, R.; Nair, S.; Eckert, C.; Clarkson, P.J. Visualization Techniques for Product Change and Product Modelling in Complex Design. In *Diagrammatic Representation and Inference. Lecture Notes in Computer Science*; Goos, G., Hartmanis, J., Leeuwen, J.v., Blackwell, A.F., Marriott, K., Shimojima, A., Eds.; Springer: Berlin/Heidelberg, Germany, 2004; Volume 2980, pp. 388–391. [CrossRef]
29. Sharman, D.M.; Yassine, A.A. Characterizing complex product architectures. *Syst. Eng.* **2004**, *7*, 35–60. [CrossRef]
30. Peterson, T. Understanding Systems through Graph Theory and Dynamic Visualization. In Proceedings of the 2014 Ground Vehicle Systems Engineering and Technology Symposium (GVSETS), Novi, MI, USA, 4–6 August 2015.
31. Bastian, M.; Heymann, S.; Jacomy, M. Gephi: An open source software for exploring and manipulating networks. In Proceedings of the Third International AAAI Conference on Weblogs and Social Media, Menlo Park, CA, USA, 17–20 May 2009. [CrossRef]
32. Batagelj, V.; Mrvar, A. Pajek—Analysis and Visualization of Large Networks. In *Graph Drawing Software*; Springer: Berlin/Heidelberg, Germany, 2004; pp. 77–103. [CrossRef]
33. Turner, D. Qualitative Interview Design: A Practical Guide for Novice Investigators. *Qual. Rep.* **2014**, *15*, 754–760.
34. Leech, B.L. Asking Questions: Techniques for Semistructured Interviews. *PS Polit. Sci. Polit.* **2002**, *35*, 665–668. [CrossRef]

35. Adams, W.C. Conducting semi-structured interviews. *Handb. Pract. Progr. Eval.* **2015**, *4*, 492–505.
36. Kallio, H.; Pietilä, A.-M.; Johnson, M.; Kangasniemi, M. Systematic methodological review: Developing a framework for a qualitative semi-structured interview guide. *J. Adv. Nurs.* **2016**, *72*, 2954–2965. [CrossRef] [PubMed]
37. Flanagan, J.C. The critical incident technique. *Psychol. Bull.* **1954**, *51*, 327–358. [CrossRef]
38. Rupp, C. *Requirements-Engineering und Management*, 4th ed.; Hanser: München, Germany, 2007. [CrossRef]
39. Mavin, A.; Wilkinson, P.; Harwood, A.; Novak, M. Easy Approach to Requirements Syntax (EARS). In Proceedings of the 2009 17th IEEE International Requirements Engineering Conference, Atlanta, GA, USA, 31 August–4 September 2009; pp. 317–322. [CrossRef]
40. Firesmith, D. Specifying Good Requirements. *J. Object Technol.* **2003**, *2*, 77–87. [CrossRef]
41. Maalem, S.; Zarour, N. Challenge of validation in requirements engineering. *J. Innov. Digit. Ecosyst.* **2016**, *3*, 15–21. [CrossRef]
42. Forrester, J.W. Industrial Dynamics. *J. Oper. Res. Soc.* **1997**, *48*, 1037–1041. [CrossRef]
43. Plattform Industrie 4.0: Details of Asset Administration Shell. 2020. Available online: https://www.plattform-i40.de/IP/Redaktion/DE/Downloads/Publikation/Details_of_the_Asset_Administration_Shell_Part1_V3.pdf?__blob=publicationFile&v=5 (accessed on 2 November 2021).
44. Available online: https://industrialdigitaltwin.org/ (accessed on 7 October 2021).
45. Available online: https://unity.com/de (accessed on 7 October 2021).
46. Albers, A.; Lohmeyer, Q. Advanced systems engineering–towards a model-based and human-centered methodology. In Proceedings of the 9th International Symposium on Tools and Methods of Competitive Engineering (TMCE 2012), Karlsruhe, Germany, 7–11 May 2012.
47. Kecher, C.; Salvanos, A.; Hoffmann-Elbern, R. *UML 2.5: Das Umfassende Handbuch*; Rheinwerk Verlag: Bonn, Germany, 2017.
48. Barlas, Y. Formal aspects of model validity and validation in system dynamics. *Syst. Dyn. Rev. J. Syst. Dyn. Soc.* **1996**, *12*, 183–210. [CrossRef]