

Article

Model-Based Requirements Management in Gear Systems Design Based On Graph-Based Design Languages

Kevin Holder ¹, Andreas Zech ², Manuel Ramsaier ² , Ralf Stetter ^{2,*}, Hans-Peter Niedermeier ¹, Stephan Rudolph ³ and Markus Till ²

¹ ZF Friedrichshafen AG, 88046 Friedrichshafen, Germany; kevin.holder@zf.com (K.H.); hans-peter.niedermeier@zf.com (H.-P.N.)

² Fakultät Maschinenbau, Hochschule Ravensburg-Weingarten, 88250 Weingarten, Germany; zecha@hs-weingarten.de (A.Z.); ramsaier@hs-weingarten.de (M.R.); till@hs-weingarten.de (M.T.)

³ Statik und Dynamik der Luft- und Raumfahrtkonstruktionen, Universität Stuttgart, 70569 Stuttgart, Germany; rudolph@isd.uni-stuttgart.de

* Correspondence: stetter@hs-weingarten.de; Tel.: +49-751-501-9822

Received: 30 August 2017; Accepted: 25 October 2017; Published: 27 October 2017

Abstract: For several decades, a wide-spread consensus concerning the enormous importance of an in-depth clarification of the specifications of a product has been observed. A weak clarification of specifications is repeatedly listed as a main cause for the failure of product development projects. Requirements, which can be defined as the purpose, goals, constraints, and criteria associated with a product development project, play a central role in the clarification of specifications. The collection of activities which ensure that requirements are identified, documented, maintained, communicated, and traced throughout the life cycle of a system, product, or service can be referred to as “requirements engineering”. These activities can be supported by a collection and combination of strategies, methods, and tools which are appropriate for the clarification of specifications. Numerous publications describe the strategy and the components of requirements management. Furthermore, recent research investigates its industrial application. Simultaneously, promising developments of graph-based design languages for a holistic digital representation of the product life cycle are presented. Current developments realize graph-based languages by the diagrams of the Unified Modelling Language (UML), and allow the automatic generation and evaluation of multiple product variants. The research presented in this paper seeks to present a method in order to combine the advantages of a conscious requirements management process and graph-based design languages. Consequently, the main objective of this paper is the investigation of a model-based integration of requirements in a product development process by means of graph-based design languages. The research method is based on an in-depth analysis of an exemplary industrial product development, a gear system for so-called “Electrical Multiple Units” (EMU). Important requirements were abstracted from a gear system specification list and were analyzed in detail. As a second basis, the research method uses a conscious expansion of graph-based design languages towards their applicability for requirements management. This expansion allows the handling of requirements through a graph-based design language model. The first two results of the presented research consist of a model of the gear system and a detailed model of requirements, both modelled in a graph-based design language. Further results are generated by a combination of the two models into one holistic model.

Keywords: graph-based design language; requirements engineering; life-cycle engineering; gear systems design; digital product development

1. Introduction

Requirements are a decisive factor in industrial product development (compare e.g., [1]). Four of ten top risks in projects are connected with requirements [2]. Analyses of industrial product development projects showed that only 52 percent of the originally allocated requirements appear in the final released version of the product [3]. Obviously, product development engineers are challenged with managing the requirements of their products, as the described research is mainly motivated by this challenge. There is a prominent need in industry for efficient and effective requirements management methods and tools. However, the complicatedness and complexity of current products aggravate such approaches. Therefore, an important research question arises: How can product development engineers efficiently and effectively manage their requirements? One possible approach can be through graph-based design languages. The unique contribution of the presented research is the in-depth investigation of the feasibility and the advantages of this kind of approach.

Graph-based languages have gained rising attention in the past years. Graph-based means that the nodes of a graph are used as an abstract placeholder for real objects and that, instead of the design object, a graph representation is processed by machines (i.e., computers). Central approaches use visual, graph-based design languages formulated in UML (Unified Modelling Language), which allow both the creation of new engineering models and the reuse of pre-existing engineering models and know-how. The use of the UML as an abstract representation of the future product to be designed allows for the consistent integration of various disciplinary domain models. The state of the art of graph-based design languages is described in Section 3.2. The research presented in this paper is lead on the example of an industrial product, a gear system for so-called “Electrical Multiple Units” (EMU), which are commonly applied in trams, suburban trains, or high-speed railway vehicles. In accordance with the product development process stage of the sample project, the research is mainly focused on stages of the product development process in which important requirements are already defined. For the sake of clarifying the scope of the research, the product development process can be described by the model in Figure 1.

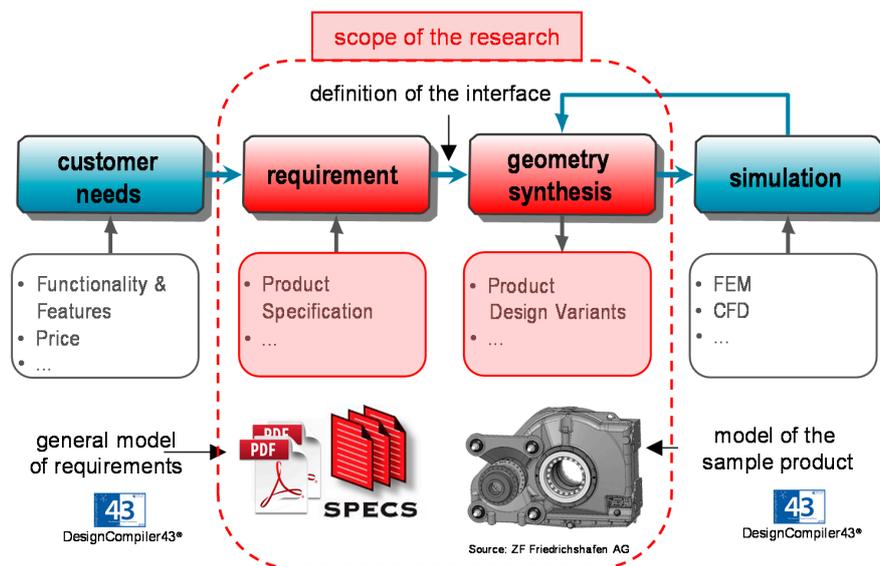


Figure 1. Clarification of the scope of the research.

This model consists of an illustration of five main process steps. In the first step, “Customer Needs” are collected and clarified. These can be transformed into a product specification that is generated in a second step, “Requirements”. Based on specific “Requirements”, a specific design approach (“Geometry Synthesis”) can be carried out. A further step of the design process is “Simulation”, and

within this step, product analysis processes are carried out by means of simulation preprocessor, solver, and postprocessor, such as the Finite Element Methods (FEM) or Computational Fluid Dynamics (CFD). It is important to note that further steps of the product life cycle are not shown in this simple model. In the presented research, the scope was on the process steps “Requirement” and “Geometry Synthesis”. There is no doubt that strategies, methods, and tools which help determine and clarify the “end-user” customer needs are also a promising field for research. However, due to the nature of the analyzed product development processes, the later steps were chosen as a focus of the presented research.

The rest of this paper is structured as follows (compare Figure 2): The research method for the investigation of the feasibility and the advantages of a model-based approach is described in Section 2. Section 3 contains the state of the art in the scientific fields “requirements management” and “graph-based design languages”. The main analytical part in the investigation, the analysis of an industrial product development process, is reported in Section 4. Section 5 describes the first research result, a model of the sample product in a graph-based design language. The second research result, a general model of requirements, is explained in Section 6. A validation is achieved by combining the model of the sample product and the general model of requirements; this combination is described in Section 7. In Section 8, the research results are discussed and an initial validation in industry is reported.

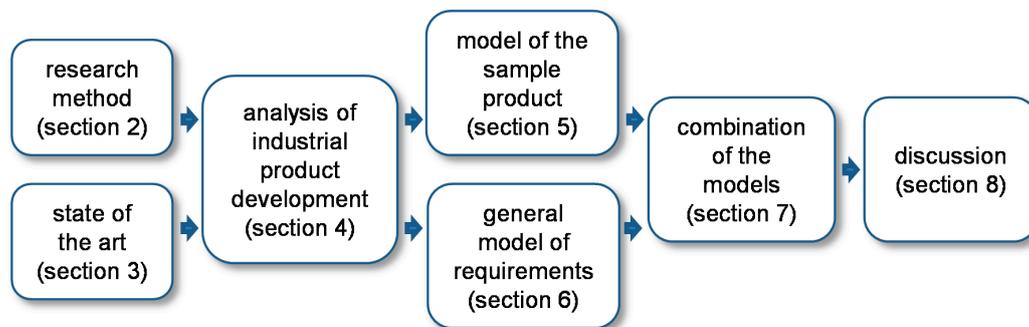


Figure 2. Structure of the paper.

2. Research Method

The described research is generally based on the Design Research Methodology (DRM) framework as presented by Blessing and Chakrabarti [4]. In this framework, four general stages of research can be distinguished (compare Figure 3).

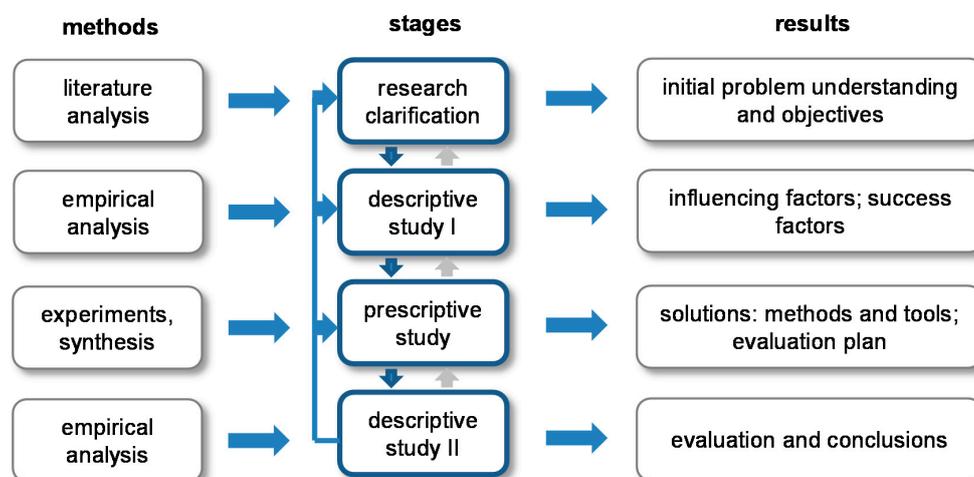


Figure 3. Four stages of the design research methodology (compare [4] and [5]).

The presented research focuses on an investigation of a novel approach (graph-based design languages for requirements management). This can be characterized as a research project type five (compare [4] and [5]). Here, product development support is created on the basis of a detailed generation of product development process understanding, and is initially evaluated. The research clarification is based on the state of the art (compare Section 3). This analysis led to the formulation of the central research thesis: It is possible to enable effective and efficient requirements engineering in industrial companies by means of the application of graph-based design languages. The first descriptive study is carried out by a detailed analysis of an industrial product development process (compare Section 4). The prescriptive study is carried out intensively and results in two models in graph-based design languages and their combination (compare Section 5 to Section 7). The evaluation and validation by means of a second descriptive study is performed in an initial manner for this kind of research project, and consists of a test of the general applicability of the approach and expert interviews in the concerned industrial company (compare Section 8). Obviously, the research type five (compare [4] and [5]) cannot really prove a research thesis, but it can indicate that the application of product development support is logically sound and carries some process improvement potential. A reflection of the research structure in the structure of this paper is shown in Figure 4.

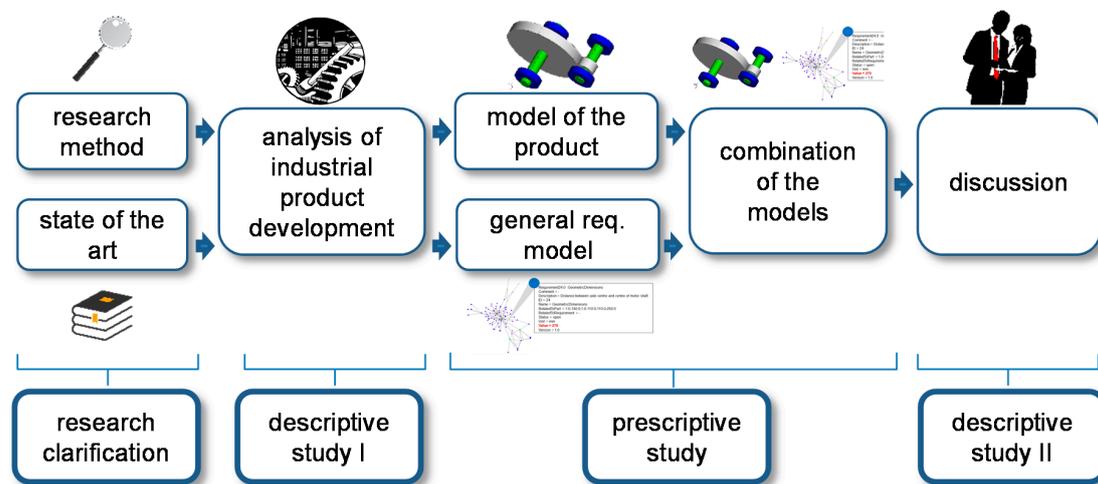


Figure 4. Research structure and structure of this paper.

3. State of the Art

This section presents the state of the art in two scientific fields, which were combined in the presented research: requirements management and graph-based design languages.

3.1. Requirements Management

3.1.1. Research Activities Concerning Requirements Management

An overwhelming amount of literature describing requirements management and requirements engineering can be found; it is therefore impossible to present a complete picture. A very good overview of earlier research can be found in Darlington and Culley [6] and Jiao and Chen [7]; current additions are shown, e.g., by Zhang et al. [8]. It is important to note that international standards which are concerned with requirements management also exist: most notable are the ISO standard 29148 “Systems and software engineering—Life cycle processes—Requirements engineering” [9] and the ISO standard 15288 [10] “Systems and software engineering—Systems life cycle processes”, as well as the recently revised quality management norm ISO 9001 [11]. Furthermore, current research addresses frameworks for product lifecycle integration [8], requirements change prediction models [12],

systematic determination of product properties [13], and requirements checklists [14]. More and more research is concerned with requirements management in industrial settings.

Requirements can be defined as the purpose, goals, constraints, and criteria associated with a product development project [12]. These requirements may range from the initial functional requirements to the detailed specifications [15]. Several classification schemes for requirements were proposed in literature (compare Bühne and Herrmann [16]). Zhang et al. [8] propose a lifecycle-view-based classification in only three categories: (1) BOL—Beginning of Life, including planning, design, and production) related requirement; (2) MOL—(Middle of Life, including use, service and maintenance) related requirement; and (3) EOL—(End of Life, including reuse, material reclamation and disposal) related requirements.

A large body of research is concerned with the fuzzy front-end of product development: the transformation of the customer needs into requirements. Closely connected are approaches of qualitative and quantitative market research, which are often referred to as voice of the customer (VOC) studies. Early research reports in the 1990s [17] already identify four main aspects: customer needs, a hierarchical structure of the needs, priorities of the needs, and customer perceptions of performance. Several research studies are concerned with the integration of the voice of the customer into the subsequent processes of product development. For this integration, a number of models, methods, and tools were analyzed, such as the Kano model [18], CAD, and TRIZ [19], as well as virtual customer environments [20]. A focus on Collaborative Product Development/Definition and Management (CPDM) solutions for sharing, archiving, and managing customer needs (amongst other product information) can be identified as well (compare e.g., [21]). Salado and Nilchiani [22] were able to establish categorization models of requirements which are based on a well-known model of human needs. The proposed categorization model supports elicitation of requirements and facilitates the detection of requirements that are not relevant to the system and of constraints that limit the solution trade-space without supporting the satisfaction of new needs.

Several authors use the notion “requirements engineering”. A number of different definitions exist which try to distinguish the notions of “requirements engineering” and “requirements management”, but these terms can be used synonymously [17]. In this paper, requirements management and requirements engineering are understood in a holistic sense as a systematic and disciplined approach for specifying and managing requirements with the following objectives:

- to identify the relevant requirements, achieve consensus between the stakeholders, document the requirements in accordance with given standards, and manage the requirements in a systematic manner;
- to understand and document the wishes and needs of the stakeholders; and
- to specify and manage requirements in order to minimize the risk that a system will not correspond to the wishes and needs of the stakeholders (compare Bühne and Herrmann [16]).

Similarly, the ISO standard 29148 [9] defines requirements management as “activities that ensure requirements are identified, documented, maintained, communicated, and traced throughout the life cycle of a system, product, or service”.

3.1.2. Stages in Requirements Management

In order to structure the later discussion in this paper, a model was developed on the basis of requirements engineering process models (compare e.g., [23,24]), which allows for distinguishing different stages in requirements management (Figure 5).

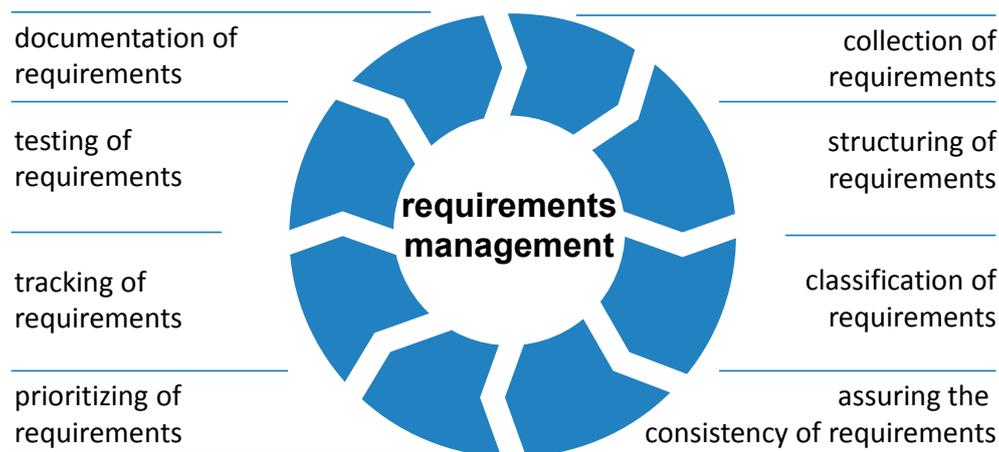


Figure 5. Stages in the requirements management process cycle.

The main stages are briefly explained in remaining part of this sub-section.

Collection of Requirements

A starting point of the requirements management process might be the collection of requirements, though this step may be repeated again and again during the process of product development. The collection of requirements may include an application of methods such as user surveys, benchmarking, or quality function deployment (QFD). In this stage, the voice of the customer plays an extremely important role (compare e.g., [19]). It is important to note that requirements specifications can never fully cover all aspects of an envisaged product [25].

Structuring of Requirements

The notion “structuring of requirements” may include activities which are intended to arrange larger lists of requirements into some kind of sensible structure. The structuring of requirements is important, because the collection of requirements for complex systems may be vast, and an overview and handling of the large set of requirements usually requires some structure, e.g., the product structure.

Classification of Requirements

Several literature sources recommend a classification of requirements; important aspects may be the general type (functional requirement, quality requirement, or structural requirement), the solution dependence (goal oriented requirements, scenario-oriented requirements or solution oriented requirements), or the abstraction level (compare Bühne and Herrmann [16]). Furthermore, Becattini et al. [14] and Salado and Nilchiani [22] also present structures for classification.

Assuring the Consistency of Requirements

Assuring the consistency of requirements includes activities which are intended to find contradictions between requirements (these contradictions need to be resolved) and to identify competitive requirements. In industrial practice, these activities are extremely important [1] and should be started early.

Prioritizing Requirements

In the case of competitive requirements, a prioritization of requirements is necessary. Here, it is crucially important to bring up conflicts and to find and document compromises, especially with respect to industrial practice [1].

Tracking of Requirements

The tracking of requirements (compare requirements traceability [26]) is a central element of requirements management. It is immensely important that the requirements can be clearly identified during the whole product development process, and that a version and change management process for requirements is in place [16].

Testing of Requirements

One might argue that the testing of requirements is a major part of a product development project and cannot be seen as part of the requirements management. It is, however, a very important aspect of requirements management that these crucial activities are supported in an optimum manner, and that the requirements are readily available in a consistent, plausible form for the analysis activities.

Documentation of Requirements

The structured documentation of requirements is a main challenge because usual requirements management systems lack an updated representation of the product structure.

3.1.3. Requirements Management in Industry

Today, elaborate software packages for requirements management are available; best known are the products Teamcenter Systems Engineering [27], Rational DOORS [28], and Serena Dimensions RM [29]. A profound overview of requirements engineering tools is given by Carrillo de Gea et al. [30]. These systems share the objective to support requirements management in distributed environments. Meanwhile, the exchange format ReqIF (an XML file format) is a widely accepted standard for the lossless exchange of requirements between different systems. Further information about ReqIF can be found in the OMG specification [31] or in [3]. For some years now, open source tools for requirements management have been made available, e.g., “RMF/ProR” [32] or the Requirements Editor RED ([33]—download under DTU [34]). Figure 6 shows the requirements of a gear system for light trains modelled in the Requirements Editor RED.

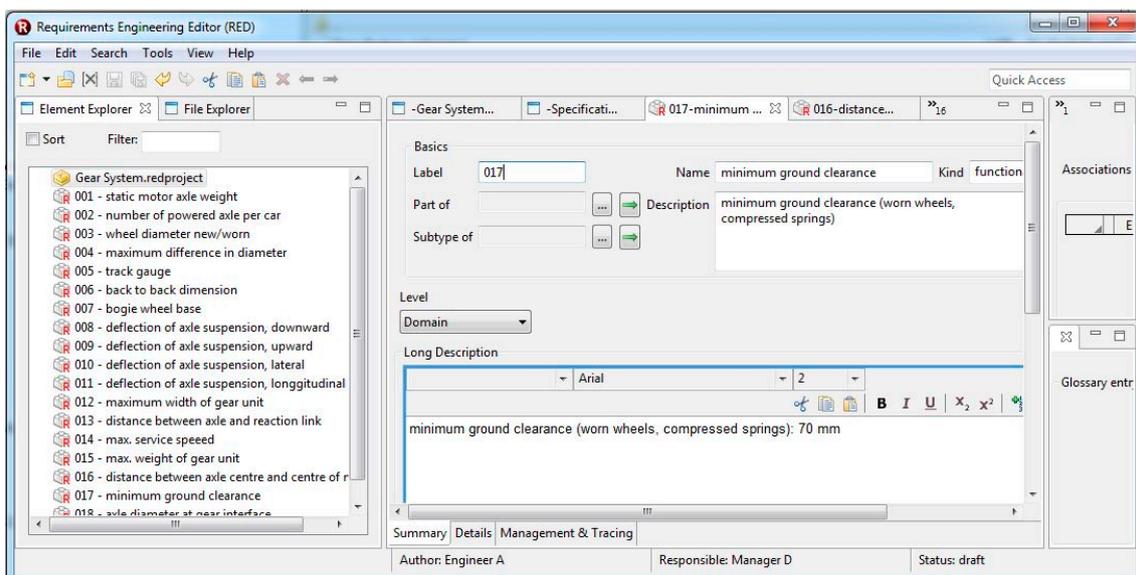


Figure 6. Requirements of a light train gear system modelled in the Requirements Editor RED.

Additionally, in industrial practice, classical office tools, such as Microsoft Word[®] and Excel[®], are used in order to support requirements management.

In the last decade, several research projects analyzed the reality of requirements management in industrial practice. Darlington and Culley [6] conducted a study in three diverse engineering companies. They formulated three key insights:

- Two fundamentally different types of “customers” need to be distinguished: the “real” customer and the “virtual” customer. The virtual customer is a representative class of individuals “who might be satisfied by some product whose design will satisfy a collection of design requirements” [6].
- Two principal identifiable roles of design requirements exist: serving as an agreement that is desired in an end product, and providing a basis from which the designer can proceed in synthesizing a solution.
- A problem/solution bias of requirements exists: the extent to which the solution itself plays a part in the expression of the design requirement can be represented in a continuum between “only problem” and “strongly solution-influenced”.

Almefelt et al. [25] studied the development of a passenger car cockpit. They were able to condense a number of concrete recommendations such as to “elaborate a concise, over-arching cross-system requirements specification providing a summary of the most important requirements”. In the analyzed project, a standardized design prerequisites document was used and perceived as a well-functioning document to present important issues for the development. A case study carried out by Sudin et al. [35] underlines that specifications represent a central element in the product development process because they provide vital information for design engineers. It also became apparent that specifications could be developed in response to the product’s usage. The vision would be a method that will identify requirements that design engineers really need to execute their task.

Li and Ahmed-Kristensen [36] report from a case study and a large-scale survey. They found out that:

- design requirements require comprehensive information from multiple sources,
- requirements for users and from users are always mixed up,
- end user requirements are critical and crucial to the success of product development,
- user sources for requirements are very complex, and
- user identification is necessary.

Borgianni and Rotini [37] were investigating amongst other facets of requirements management in the application of the Kano model for this purpose, and found out that it can shed light on divergences between different stakeholders.

In the presented studies, no special focus on the digital mapping of requirements to the product model was given. This aspect will be addressed with an analysis of the situation in an industrial company (compare next section) and the proposal of model-based requirements management (Section 5).

3.2. Graph-Based Design Languages

The authors are working on a superordinate research project. The aim of the research is the holistic integrated digital description of the product life cycle. The product life cycle includes many technical disciplines (e.g., requirement management, design, engineering, simulation, production, costing, maintenance, etc.) and therefore requires cooperation from many people and the interaction with many software tools. On the technical side, this integration process is currently focusing on complex PLM software systems which aim to achieve the integration. In some partial technical disciplines, immense efforts are realized by research and industry on the part of software development in order to promote this integration. These immense efforts result in numerous publications addressing the product modelling issues. Even graph-based approaches were analyzed since the early nineties (compare Finger and Rinderle [38]). Several research publications focus on product life cycles, such as Le Duigou and Bernard [39], and on integrated product and process models (for an excellent overview, see Eckert et al. [40]). The distinctive approach in this project is the application of an engineering

framework of graph-based design languages, which allows the reuse of pre-existing engineering models and know-how.

The concept of graph-based design languages (see e.g., Rudolph [41]) has evolved over the last ten years into a generic framework for the definition of computerized design processes. Generally, a graph is a representation of a set of objects where some pairs of objects are connected by links. The interconnected objects are represented by mathematical abstractions that can be called “nodes”, and the connections are as well represented by abstractions that can be called “edges”, which may be directed or undirected. On the basis of prior work, Rudolph [42] developed a consistent system to represent geometry and other forms of information using graphs, and has built a graph-based design language. In principle, design languages consist of a meta-model that stores all relevant parametrical and topological design information and acts as a centralized model repository [42].

These graph-based design languages are realized with UML (Unified Modelling Language). UML is the primary modelling language in computer science. Computer science is known for its very fast development cycles, which require very powerful abstract description forms (modern graphical modelling languages) for the software development process and highly automated development tools to achieve these extremely short innovation cycles. UML has been developed by software engineers and provides many features to model data in order to describe object-oriented software [43]. The latest version of UML 2.5 was released in June 2015.

It is one main hypothesis in this project that the product development processes for conventional projects can greatly profit from the basic ideas and theories behind these automated development tools. The basic structure of the application of the graph-based design language selected in this research project is illustrated in Figure 7.

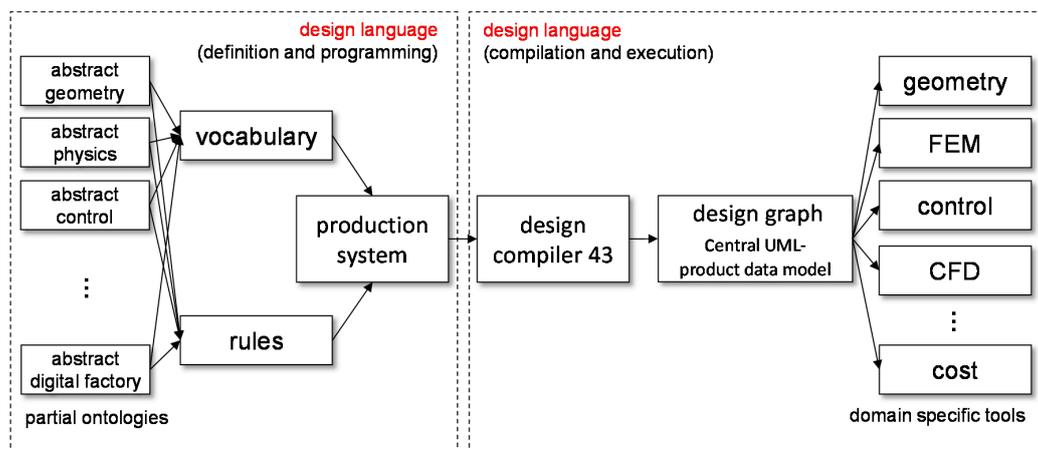


Figure 7. Application of a graph-based design language (compare Rudolph [44]).

In this graph-based design language, the engineering objects are represented by UML classes (the vocabulary of a design language), and model transformations are represented by rules, i.e., the grammar of the design language. In a so-called “production system”, the rules are specified by the engineer and later executed in order to instantiate the vocabulary classes. Whereas the UML class diagram can be seen as a blueprint of all possible products (e.g., all multi-stage planar gear systems), the design graph resulting from the execution of the production system represents a specific product (e.g., gear system for customer x with two gear stages). This execution process builds up the central data model, also referenced as the “Design Graph”. From this high-level central data model, different interfaces generate domain-specific models, such as geometry models or simulation models (CAD, CFD, FEM, etc.) fully automated with no user interaction. One main advantage of this central data model consists in the information content, which is not limited to geometry definition, but can also include abstract information concerning physical loads, materials, functions, or other

aspects. By this approach, it is possible to address a multitude of programs in engineering product development processes. The central benefit of graph-based requirements management based on UML is the consistent interconnection between structural, geometric, functional, and physical information.

One important advantage in this project is the availability of the “DesignCompiler43”. This compiler is developed by the IILS mbH (<http://www.iils.de>) in cooperation with the University of Stuttgart, which is one of the cooperation partners in this project. DesignCompiler43 is an eclipse-based software tool for the compilation of design languages formulated in UML. The design compiler can process design languages on all abstraction levels to the full detail of the simulation models. This leads to a fully automated process within a unified framework. The work on DesignCompiler43 is ongoing, but many important functionalities are already available in the current version.

The application of a graph-based design language and the usage of UML diagrams in this process is explained in Figure 8.

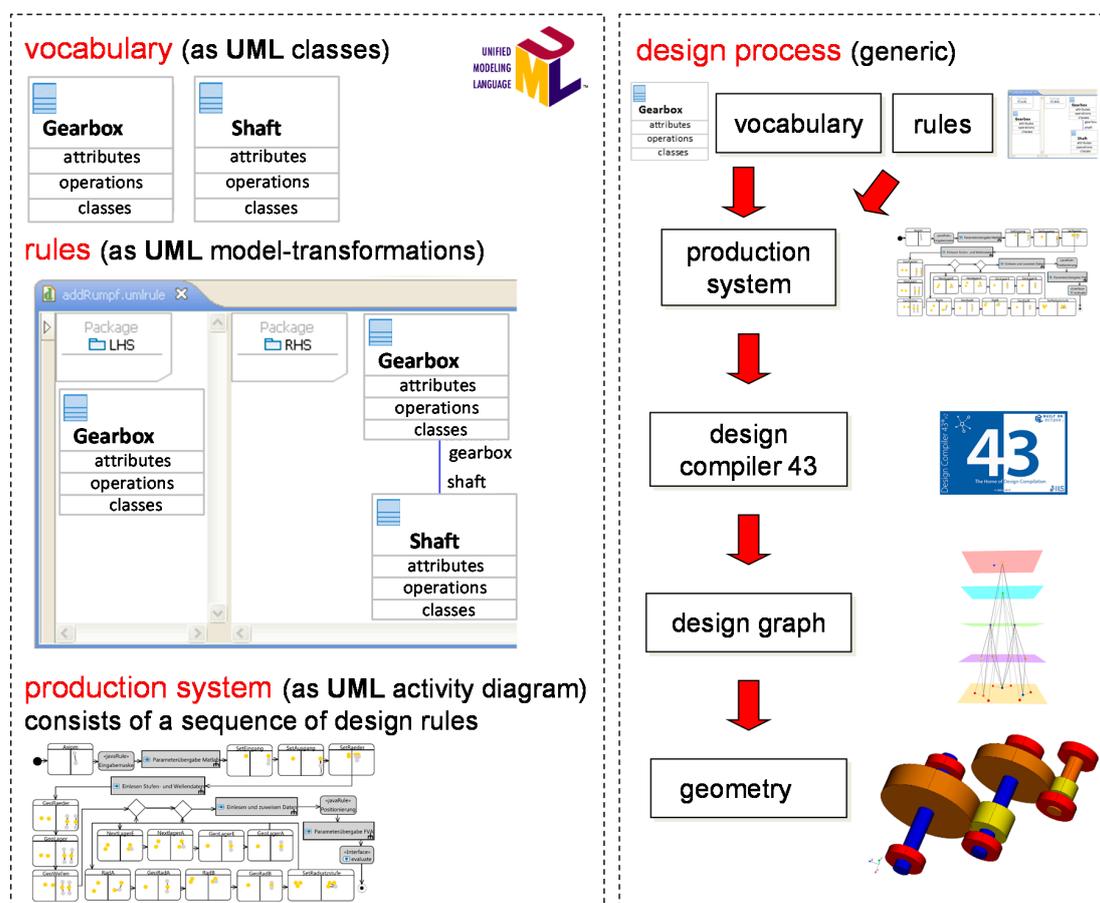


Figure 8. Application of a graph-based design language and usage of Unified Modelling Language (UML) diagrams.

As stated above, the main elements of a graph-based design language are vocabulary and rules. Vocabulary are UML-based components, e.g., the Gearbox itself or one of its shafts (Shaft). Rules are UML model transformations: these show the input status (only car) on the left side (LHS) and the output state on the right side (LHS—Gearbox with connected Shaft—see Figure 1). The engineer creates the UML model, which compiles the program into a design graph, from which, for example, the geometry of the product can be derived. Vocabulary and rules are linked in a so-called “production system” (UML activity diagram). Through the design compiler, the rules are executed to instantiate the class/vocabulary. This compilation process builds the central data model. From this central data model,

different interfaces generate domain-specific, generic models, such as geometry models or simulation models. This application of graph-based languages allows for the automatic generation and evaluation of multiple product variants and is currently being introduced in several industrial companies.

4. Analysis of an Industrial Product Development Process

This section presents the results of the first descriptive study in the research, the analysis of an industrial product development process.

4.1. Example Product: Railway Vehicle Gearboxes

The sample product is a gearbox for electrical railway vehicles (Figure 9). Such gearboxes are mounted in (pivoted) bogies of rail vehicles (compare also a 3D view in [45]); typically there are several driven axles per vehicle.

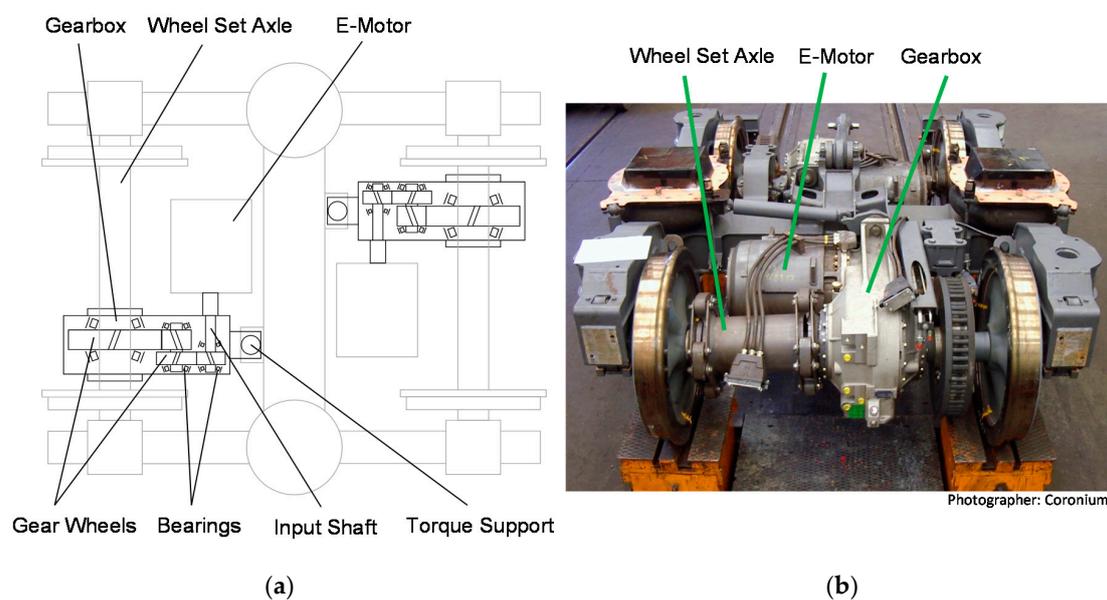


Figure 9. (a) Schematic top view of a bogie, main components of railway vehicle gear system; (b) picture of a bogie; photographer: Coronium.

The gearboxes are “single-speed reducers” with no possibility to interrupt the powertrain and are developed for different categories of EMU. The gearboxes are equipped with typically 1 up to 2 gear stages of steel gear wheels if the input and output shafts are on one plane, and up to 3 or more stages if the input and output shafts are not on one plane, where the directional deflection is done with a bevel gear or a hypoid gear stage. Further subdivision can be done according to the type of mounting in the bogie. Either a fixed connection to the wheel axle and a flexible coupling with the electrical motor shaft is possible, or a fixed connection to the electrical motor shaft and a flexible coupling to the wheel axle. On top of that, special solutions and mixing of both types are implemented in real products. In Figure 8, a two-stage gearbox suspended on the wheel axle with a flexible coupling to the motor shaft is shown.

In the respective company (ZF Friedrichshafen AG), the entire development process with its sub-steps is defined, therefore several company-specific instructions and templates are available. For the development process, state of the art development tools (CAD, FEM, . . .) are applied. The main focus of this investigation was the early process part, which takes place before an actual order or a “Letter of Intent” (LoI) is issued by the customer of the gear system (“Offer to Order process” (OtO)). After the order or LoI, a well-structured stage-gate process is carried out, but even before the product concept has to be developed in a considerate level of detail. This stage is especially challenging, as the basic decisions in this stage can influence the whole success of the development project.

4.2. Requirements Management in the Industrial Company

The described research project presented in this paper started with an in-depth analysis of the requirements management process of the company that develops and produces the sample product.

At the beginning of the OtO, a technical questionnaire can be used, which helps to define the basic parameters for a customer project. On this basis, the design process can start and an offer can be created and submitted. On the technical side, function parameters in industrial practice can be listed as a basic specification. These parameters can be gear ratio, maximum weight, and geometrical dimensions. In the case of an order or further development, this kind of specification can be used for project planning. The specification is a textual description of the requirements, which concern the environmental conditions of the product and product properties. In addition to the textual descriptions, the environment or interfaces are supplemented by non-textual descriptions, such as drawings and sketches or 3D models of interfering contours. The specifications often include explicitly or implicitly the definition of solving principles or product details. This restriction on certain solutions is mostly based on the experience of predecessors or similar products. The analysis of the specification is done by engineering knowledge (experience). Each related individual clause (each sentence) is evaluated. The evaluation is carried out by a "Clause by Clause" (CbC) analysis in categories, such as "accepted", "not accepted", "partly accepted", "non-applicable", or additional comments for information. Based on the CbC analysis, a coordination and negotiation process with the vehicle manufacturer (customers) can take place. A scope statement is created and a "project drawing" can be generated. For the customer, the technical solution "transmission with gear wheels" is usually out of the question, so the scope statement contains only the main technical parameters of the transmission (as design criteria e.g., standards and design torque, . . .) and the installation drawing (or/and 3D installation model) which contains e.g., materials or the rough housing shape. The configured gearbox in combination with the experience of previous products is used to calculate the product quotation. The development process with its gates and milestones starts during that part of the initial stage.

The analysis of this industrial product development process was concluded with the creation of a requirements database, which can be one initial condition for the development of models for requirements management with graph-based design languages (compare Section 6.1). The content of the database was entirely extracted through CbC analysis from a textual description provided by a customer.

5. Development of a Gear System with a Graph-Based Design Language

This section describes the first core result of the research: the composition of a model of the gear system variants during the OtO in a graph-based design language. The first main condition for the approach to connect requirements with the product is a product model with all domains, which are directly specified or necessary to reach the specified target. In this case, it is a product model created by a graph-based design language. At the current stage of research, the development of this model of gear systems is still ongoing. Figure 10 shows a scheme of the creation process of the model.

Each model that is modelled in the graph-based design language consists of various classes that are determined by decomposition of a given product model. Based on the UML notation for object-oriented systems, all derived classes can be connected to each other through utilization of association or generalization (compare Figure 4). The linked classes form a class diagram (synonymous for vocabulary in Figure 7), which represents the product architecture. Consequently, the class diagram shown in Figure 11 is valid for single-speed reducers, as they are common in EMU.

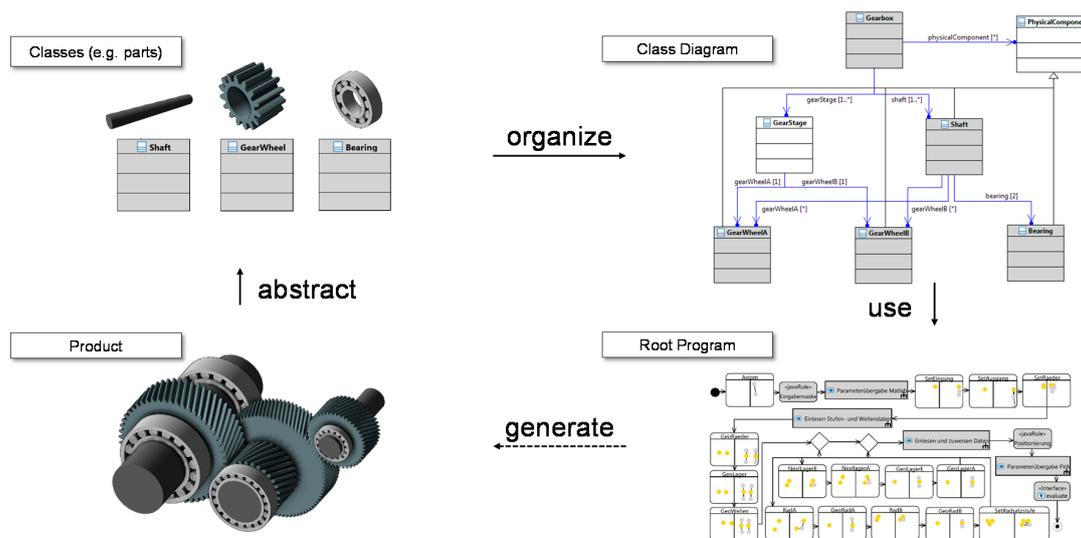


Figure 10. Scheme of the development process of a graph-based model for gear systems design variants.

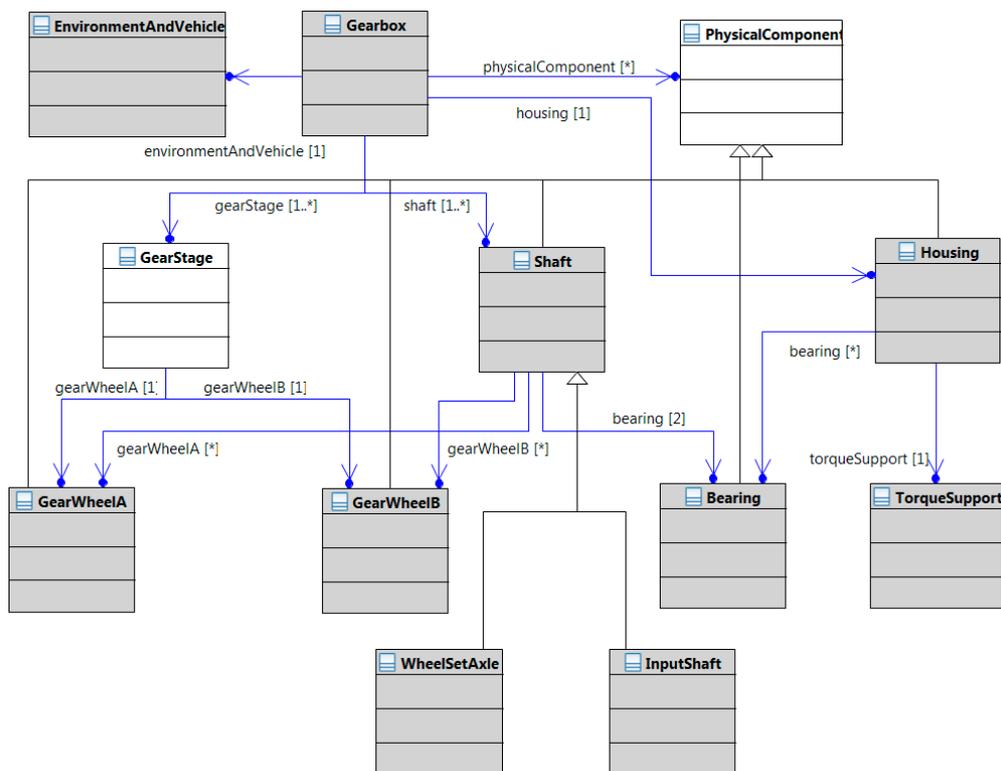


Figure 11. Simplified class diagram of a model for generating gear systems design variants.

For this early phase, only the architecture of the basic components is defined. In this case, it is the combination of the gear set and the gearbox housing into a whole “Gearbox”, whereas the gear set can be composed of any number of stages, each consisting of two gearwheels. For the gearwheels, a specialization into a driving wheel marked with “GearWheelA” and driving wheel marked with “GearWheelB” is implemented. The wheels are mounted on shafts (“Shaft”); the connection to the “Housing” is done by the rolling “Bearing”. In addition, a specialization of shafts for the input and output shafts is also carried out (“InputShaft” and “WheelSetAxle”). Classes which represent a gearbox component inherit properties of the abstract class “PhysicalComponent” for a geometric representation.

Combined with the class diagram shown, the separate root program (synonymous for rules in Figure 7) leads to a virtual model. The root program contains the steps necessary to build a detailed variant of a gear system and to check this solutions on the basis of the collected requirements. The rules lead to the instantiation of classes, which give the expression of various variants of the gear system. In general, the class diagram becomes provided with logic through the rules within the root program.

In Figure 12, instances of one typical variant of gear system (depending on requirement inputs) are displayed geometrically (on the left side) and represented by a pronounced design graph (on the right side).

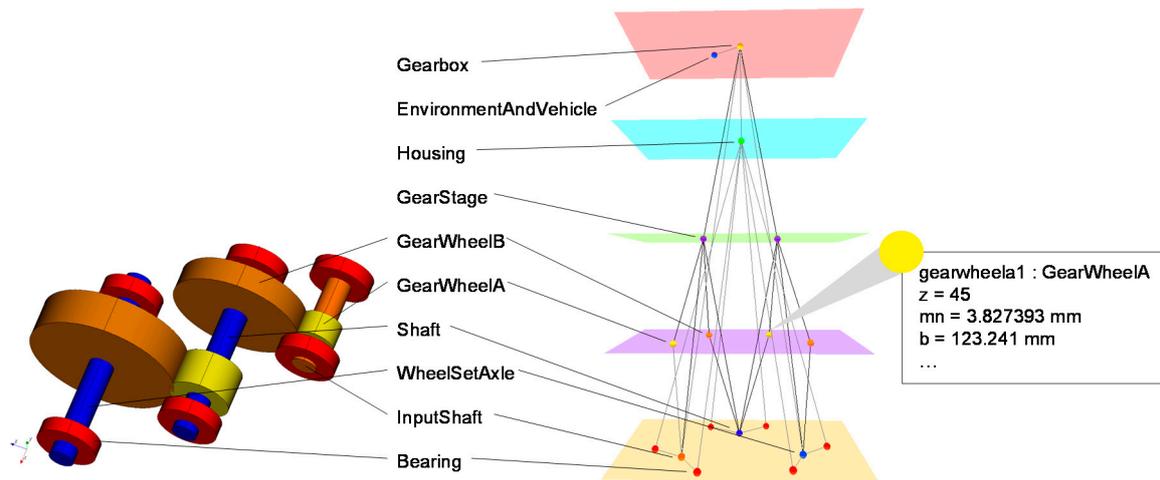


Figure 12. Graph of instances which represent the components of one typical variant of gear system (Instance attributes: z = number of teeth; mn = normal modul; b = width).

6. Requirements Management with a Graph-Based Design Language

This section presents the second core result of the described research: the description of the innovative approach towards model-based requirements management. It describes the composition of a design language for requirements management.

6.1. A General Model of Requirements in a Graph-Based Design Language

One main condition for the approach to connect requirements with the product, which is shown here, is a requirement model. In this case, it is a requirement model created by a graph-based design language (Figure 13).

The class diagram shown in Figure 13 consists of a main class named “Requirement” that includes attributes ID, Name, Description, Version, Status, Unit, Value, Comment, RelatedToPart, and RelatedToRequirement (will be used for structuring requirements). Each of the requirements in the database that was created from requirements in the company (compare Section 4.2) was taken into account and was illustrated by model in the graph-based design language. Consequently, these requirements could be shown as “Requirement” instances within the design graph after the compilation of the model (compare Figures 10 and 12). According to Ammenwerth [46], requirements are supposed to have further attributes such as “evaluation criterion”, “possible characteristics”, “desired characteristics”, “weighting” (prioritization), and “actual characteristics”. Assigning content values to these attributes has to be done by engineering knowledge (experience) during the requirement collection phase.

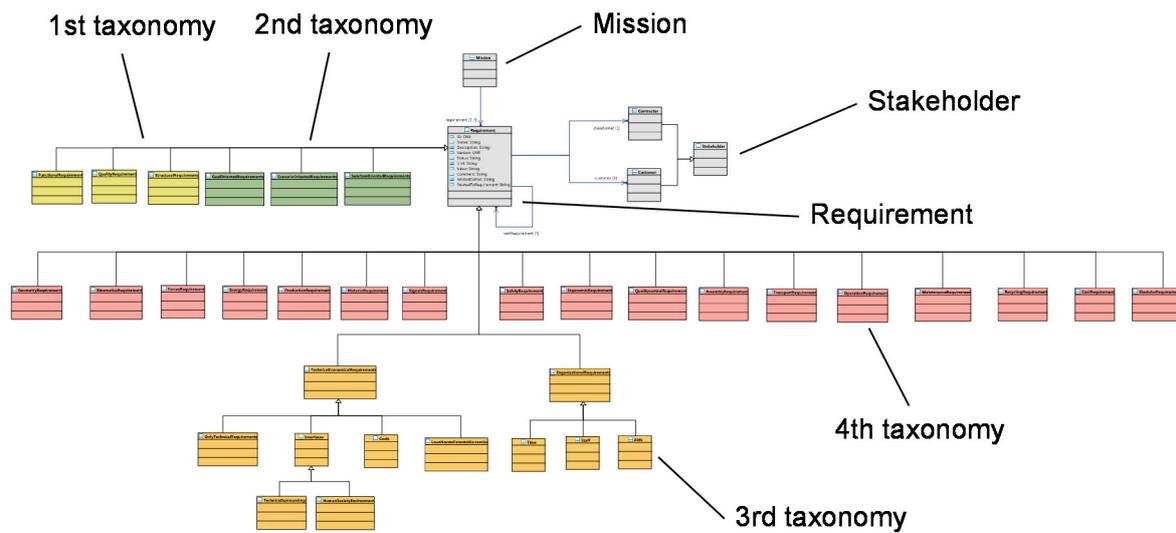


Figure 13. Implemented class diagram for classification and structuring of database requirements.

Furthermore, Figure 13 contains a class “Mission” that has an association to class “Requirement”. “Requirement” is associated to “Stakeholder” with classes “Contractor” and “Customer”. Both inherit from “Stakeholder” by specialization.

In contrast to the model for gear systems, in which classes are determined by decomposition of a given product model (compare Section 5), the classes for requirements management are determined according to taxonomies which can be derived from literature (compare Section 4.1). That leads to a class diagram which contains several approaches for classifying requirements (Figure 13; colors represent taxonomies) additional to the general description of requirement entities. The classification of requirements in model-based requirements management is in general similar to existing requirements management systems.

Until now, four different taxonomies are implemented (compare Figure 13). The first taxonomy considers the aspect of general type, implemented by the yellow classes “FunctionalRequirement”, “QualityRequirement”, and “StructuralRequirement”. The second taxonomy takes the solution dependence into account by implementation of green domains “GoalOrientedRequirement”, “ScenarioOrientedRequirement”, and “SolutionOrientedRequirement”. According to Figure 8, requirements can be classified by a third taxonomy, which is shown by orange colored domains (“OnlyTechnicalRequirement”, “TechnicalSurrounding”, “HumanSocietyEnvironment”, “Costs”, “LawsNormsPatentsWarranties”, “Time”, “Staff”, and “Aids”).

Beattini et al. [14] propose a further taxonomy (“Geometry-”, “Kinematics-”, “Forces-”, “Energy-”, “Production-”, “Material-”, “Signals-”, “Safety-”, “Ergonomic-”, “Qualitycontrol-”, “Assembly-”, “Transport-”, “Operation-”, “Maintenance-”, “Recycling-”, “Cost-”, and “SchedulesRequirement”), which is mostly utilized in checklists for conceptual design (highlighted by red color in Figure 12).

The vocabulary in Figure 13 includes all four taxonomies. However, the experience during the modeling of the gear system (compare Section 7) shows that the taxonomy introduced by Beattini et al. [14] seems to be most appropriate for classification within the described design context. Further work will also look into further taxonomies, such as the categorization model presented by Salado and Nichani [23].

A further purpose of a model of requirements in a graph-based design language can be the structuring of requirements (compare Section 4.1). In order to structure requirements, all taxonomy classes are linked by specialization with the main class “Requirement”. Thus, all taxonomy classes inherit the attributes of “Requirement”. As shown in Figure 12, “Requirement” has an aggregation on itself. Hence, the attribute “RelatedToRequirement” is able to be implemented as well.

In addition to structuring requirements according to attributes, requirements may also be structured polyhierarchically. A polyhierarchy corresponds to a general detailing of requirements. Basically, a polyhierarchical requirements model is able to be represented by a design graph (Figure 14).

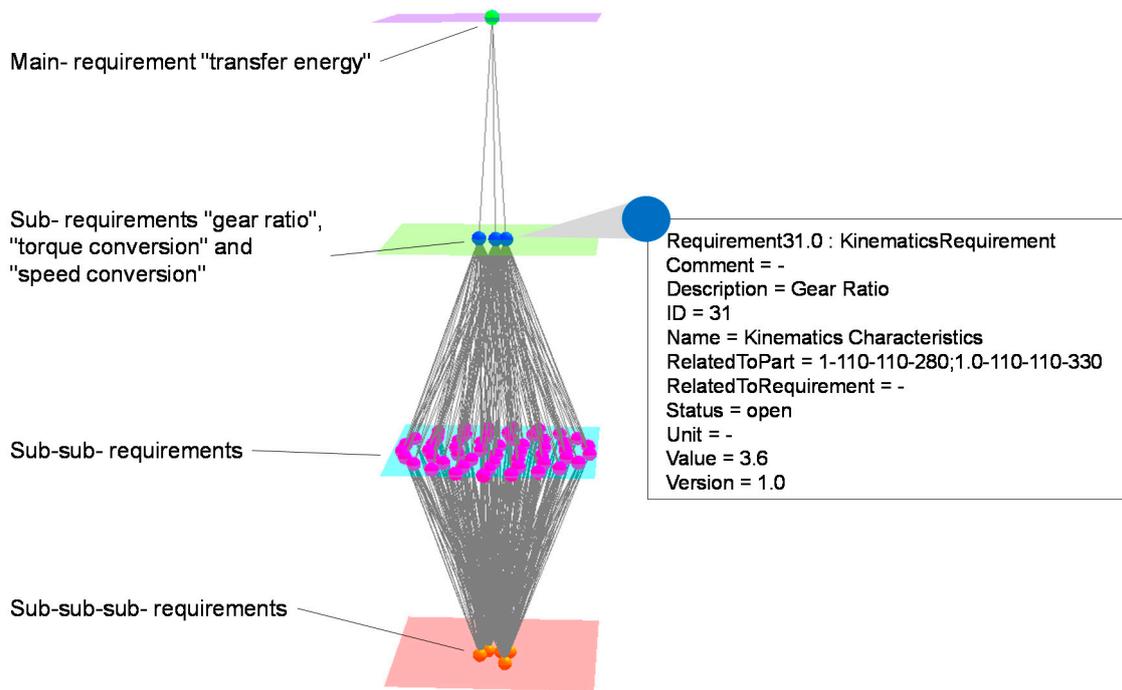


Figure 14. Polyhierarchy of requirements from the database.

The polyhierarchical requirements model may be objected to the application of structural and content-related quality criteria, as well as subsequent analyses such as clustering, dependency analysis, or missing requirements analysis.

The general purpose of polyhierarchical requirements structuring already was declared by Becattini et al. [10], through defining the characteristics (in this context synonymous to quality criteria) that a design specification should have. They listed the characteristics validity, completeness, operability, non-redundancy, conciseness, and practicability.

To achieve a polyhierarchical requirements model, the requirements database from the sample product (compare Section 4.2) has been divided into main-, sub-, sub-sub-, and sub-sub-sub-requirements. The main requirement of a gear system, “transfer energy”, is located on the top layer (compare Figure 14). In general, the main requirement equals the root of the graph which represents the overall strategic goal; Martin [47] calls it “Mission”. The second layer contains three sub-requirements that describe the “gear ratio”, “torque conversion”, and “speed conversion”. The lower two layers, sub-sub-requirements and sub-sub-sub-requirements, maintain the remaining database requirements. Figure 14 shows the polyhierarchical graph where, for instance, one of the instances of database requirements is highlighted (blue color). The requirement with ID #31 was classified as “KinematicsRequirement” and structured as sub-requirement. Furthermore, content values from requirement database are attached to attributes (e.g., Description = Gear Ratio, Name = Kinematics Characteristics, Value = 3.6, etc.).

6.2. Application Possibilities of the Model of Requirements

In this subsection, application possibilities of the general approach of a model-based requirements management process are explained together with their potential advantages. The structure of the subsection follows the model shown in Figure 5.

6.2.1. Application in the Collection of Requirements

Beccattini et al. [14], amongst others, point out that completeness is an important characteristic of sets of requirements for a product. Consequently, the collection of requirements is a crucial step in requirements management. As stated in Section 4.1, main sources for requirements are methods like marketing studies, product clinics, benchmarking, and quality function deployment (QFD). Additionally, earlier products are a main source for requirements, because many products are evolutionary improvements of earlier products, e.g., in automotive industry (compare Almfelt et al. [27]). The general approach of model-based requirements management allows to document the requirements in a strong relation to the sub-systems respective modules of the earlier product. Consequently, this approach can support the extraction of requirements from earlier products that can be reused because the same or similar modules will be used in a future design. Additionally, the stringent logic in model-based requirements management will ease the identification of the crucial requirements in the earlier product models. The developed requirement models of earlier products, which are formulated in graph-based design languages, can also be analyzed in multiple ways. Beccattini et al. [14] were able to show that any tools which aim at stimulating reflections can support the generation of non-obvious requirements. Such human reflection can be fostered with requirement models of existing products, but they can also be analyzed by means of search algorithms.

6.2.2. Application in the Structuring of Requirements

The handling of a large set of requirements usually requires some structure, e.g., the product structure. Model-based requirements management allows combining the requirements to the product structure, thus providing a straightforward way to structure requirements.

6.2.3. Application in the Classification of Requirements

The taxonomies mentioned in the literature (compare Section 3) were used as a basis for the class diagram of the design language for requirements management. Further work will also look into further taxonomies, such as the categorization model presented by Salado and Nichani [22].

6.2.4. Application for Assuring the Consistency of Requirements

A requirements model, which is integrated in a graph-based life cycle spanning product model, can offer multiple ways for exploring the product and the consistency of requirements. One promising approach is a dependency analysis between the requirements themselves, and also between product and requirement. Firstly, such analyses may allow identifying interdependent requirements which may be contradicting or competitive. Secondly, such analyses may offer the possibility to identify central and isolated requirements as well as active and passive requirements.

6.2.5. Application for Prioritizing Requirements

Bernard and Irlinger [1] were able to identify the activities which serve for prioritizing competitive requirements as crucial for the success of leading producing companies. Frequently, this prioritization requires detailed knowledge about rather concrete properties and performance parameters of the future product or system. Systems which are based on graph-based design languages can assist the generation of such knowledge, for instance, by means of sensitivity analyses. For instance, the system for the early phase of the development of gear systems (compare Section 3) allows an automated or semi-automated generation of conceptual product variants and their analysis and evaluation by means of state-of-the-art simulation tools. Such simulation results for multiple solution variants can present an excellent basis for discussions leading to well-founded systems of priorities for product performance parameters and properties. Graph-based design languages can support preliminary studies to rather concrete levels of product descriptions, but still allow stepping back to the conceptual levels, because of the efficient model integration and efficient simulation processes of multiple solution variants.

6.2.6. Application for the Tracking of Requirements

Morkos et al. [12] point out that requirement change propagation, if not managed, may even lead to product development project failure. They were able to show that more than half of a system's requirements will change before project completion and that they have a significant influence on the product development process. It can be concluded that for non-trivial product development processes, a conscious change management of requirements is mandatory. Similar to requirements management, this change management for requirements concerns people, processes, and systems. From the system viewpoint, a requirements management system must at least have unique identifiers for each requirement and must allow and ease the tracking at all phases of the product life cycle. The model-based approach that is integrated into a life cycle integrated model offers an optimal basis for this endeavor; the concrete implementation of a change management system for requirements will be an important topic for the next research steps.

6.2.7. Application in the Testing of Requirements

The testing of the fulfilment of requirements is carried out by all kinds of analyses, which determine the product performance and is usually not considered to be an integral part of requirements management. However, strong logical interrelations can be observed. Frequently, analyses of central product performance indicators are needed in order to determine the general feasibility of important requirements. Such analyses can be referred to as preliminary studies, explain certain requirements, and may lead to new requirements. For nearly all simulations and tests, requirements are the basis for the evaluation of the performance of the product. Additionally, in the case of competitive requirements, certain kinds of analyses, which can be also referred to as detail studies, may be needed in order to determine the optimum trade-off between the competitive requirements (it is important to note that in most cases, no mathematical optimum can be achieved or proven). In such cases, sensitivity analyses can be carried out in order to gain knowledge which, consequently, certain shifts of requirement levels may have. A graph-based design language allows a logical depiction of these interconnections between product analyses and requirements, thus enhancing the transparency and consequently the stability of the product development process.

6.2.8. Application in the Documentation of Requirements

The central element concerning the documentation of requirements equals the documentation in the digital product model, which is based on graph-based design languages. In the future, a link to function models can also be established. Furthermore, the possibility to link requirements to abstract physics, and thus allowing simulation processes to be connected to the question they seek to answer, can be achieved. The analysis reported in subchapter "Application in the documentation of requirements" also led to the insight that several requirements cannot be formulated with parameters (not-textual requirements) but consist of more or less detailed drawings, for instance of the possible installation space of the gear system. With current requirements management systems, these requirements can either be used as additional information without logical integration of the individual dimensions, or it can be attempted to create a number of descriptive parameters. Model-based requirements management can offer the possibility to create a requirements geometry which completely represents the given information and can be analyzed in depth (by using structural and content-related quality criteria). This requirement geometry, which may for instance allow comparisons with the product, is currently under development.

7. Combination of the Models

This section deals with the core innovative approach of the described research. From the last two sections, both conditions for the approach to connect requirements with the product are satisfied, and a model for generating railway vehicle gear systems as well as an innovative model for requirements

management were developed. In these two sections, no special focus was on the explicit digital mapping of requirements to the product model. For this kind of mapping, the definition of an interface between both models (“requirements”–“geometry synthesis”, compare Section 1) is needed. This aspect will be addressed in this section.

The term “interface” may also understood as “combination”. A combination of both models can lead to model-based requirements management. The model-based requirements management has in very general terms three main aspects which could also be understood as dimensions (Figure 15).

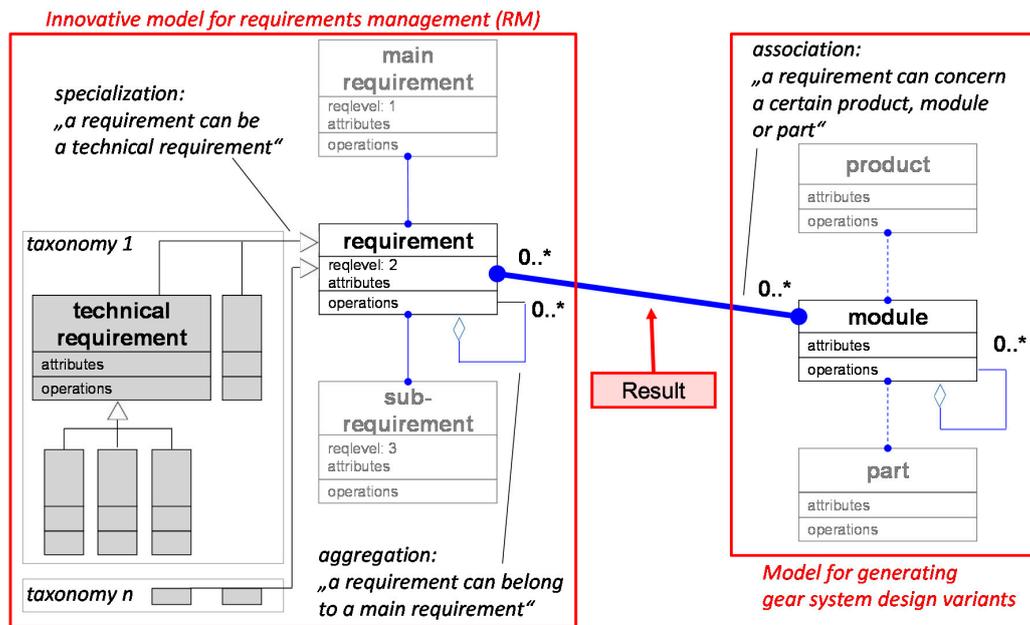


Figure 15. General approach of model-based requirements management.

The first dimension equals the model for generating gear system design variants (compare Section 5, Figure 15, right red frame), the second dimension equals the model for requirements management (compare Section 6, Figure 15, left red frame), and the third dimension equals the main approach (objective) of this research, the linking of requirements to the product graph. This linking is represented by a blue association link (Figure 15, between red frames, Result) that forms the connection between both models.

Requirements can be linked to modules or components of the product or to the product as a whole, regardless of the point at which the product life cycle is situated. That leads to an assignment of gear system classes to requirements or vice versa (compare Figure 15, blue association between red frames). Hence, attribute “RelatedToPart” is introduced within structuring of requirements, and a concrete assignment from a certain requirement to its related product, module, or part can be described.

Figure 16 shows an example for the general approach of model-based requirements management.

A connection exists for an instance of the class “requirement” that specifies the distance between axle center and center of motor shaft (compare Figure 16, highlighted blue node) to the related part of the gear system, instance of class “Housing”. Whereas the instance of the class “Housing” has an influence on its underlying components/parts.

Consequently, changes in the requirements entity (compare Figure 16, value in red) lead to changes of the product (here: causes topology changes of the resulting gearbox). The connection between requirements and the design process can be expanded. A more advanced mapping of requirements to components and assemblies will be implemented in near future.

The assignment process (instances of “requirement” to “geometry synthesis” or vice versa) is currently carried out by using search words. This kind of search has been defined and programmed

for the case of rather obvious requirements. In order to achieve appropriate general validity, it is conceivable to use an intelligent search algorithm here. Also, the concrete value of the requirement is passed through the content of instances to which a connection exists. Thus, the requirements are used for the layout process of the product variant and furthermore to a consistency between product variants and its requirements.

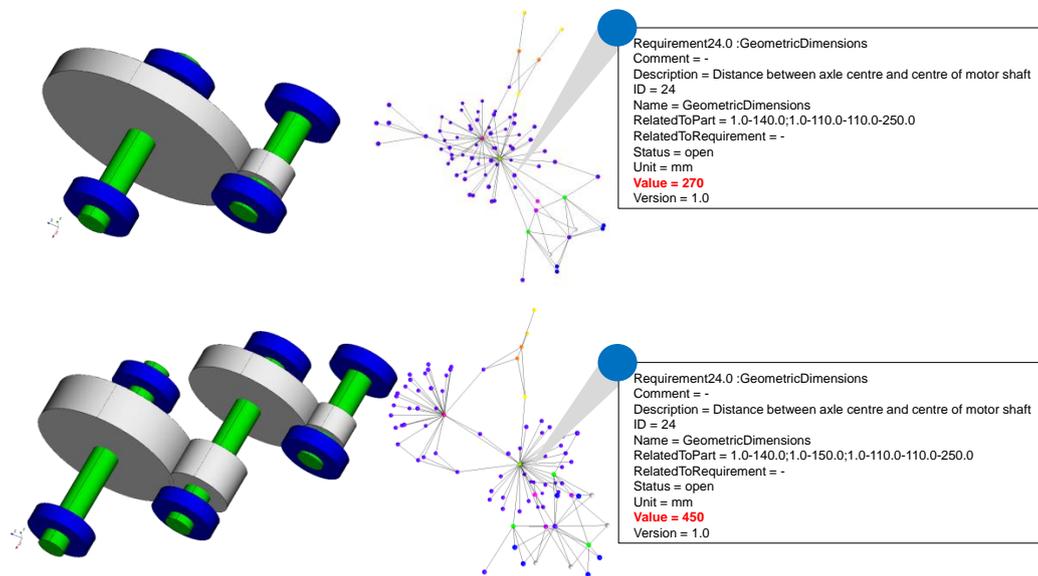


Figure 16. Example for a mapping process of requirements to related product components/parts.

Since that mapping process of requirements has happened gradually in the design process, design decisions are carried out based on the requirements. As an example, the gear ratio has an influence on the topology of the resulting gearbox. One may note that requirements can be mapped to all layers of the product architecture. Figure 16 shows an example of how requirements can be mapped to the product architecture. The product as a whole consists of five layers in the upper area of Figure 17. The layers below represent the classification of a requirements collection by using the taxonomy of Beccattini et al. [14] within Figure 13.

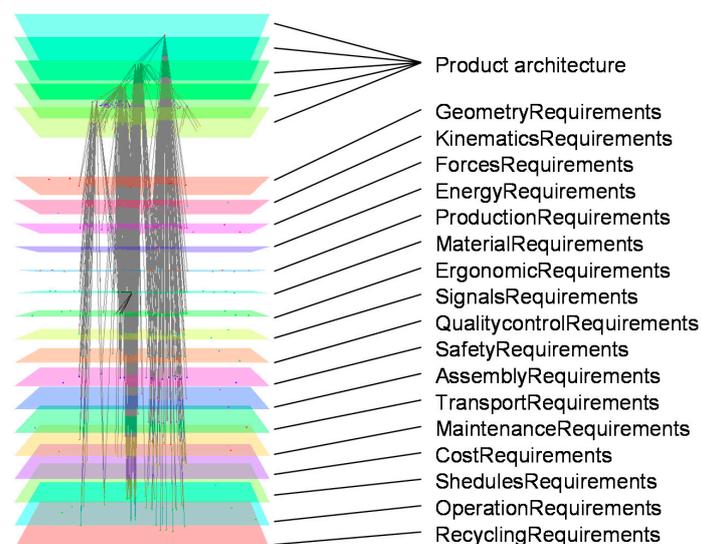


Figure 17. Example for a design graph of instances of the product architecture and requirements classified by the fourth taxonomy of Figure 13.

This structure allows depicting the product logic in the abstract domain of requirements, thus improving the logical interconnection between product components and respective requirements.

8. Discussion

This paper aims at demonstrating a mapping of model-based requirements management to a development process of gear systems by means of applying graph-based design languages. The conscious connection of specific requirements to their related product components/parts can be considered as the main contribution of the presented research.

The model for gear systems consists of a vocabulary that was determined by the decomposition of a given product structure. The use of design languages with their central data model enables generating various product designs (depending on different input requirement) in a short period of time in a fully automatic manner. Thus, the whole design space is not limited to a few, manually chosen by human, design configurations, but includes all possible variants. By downstream evaluation processes, e.g., pareto-optimal, the most suitable approach is able to be selected.

The model-based requirements management is based on a textual requirements specification of an EMU from an industrial customer. The purpose of the model is the collection, structuring, classification, assuring of consistency, prioritization, tracking, testing, as well as documentation of these requirements. Therefore, they are captured and are transferred into an UML model which results are represented by a design graph.

A further result of the research is the definition of an interface between both models.

The three results were successfully used in order to model the requirements of a real gear system family by means of the application of graph-based design languages using UML. It was possible to compile the model in the design language and to achieve a geometrical configuration. Consequently, the general realization possibility of the presented approach could be shown in an industrial context.

The results presented in Section 5 to Section 7 were discussed with development engineers in the cooperating industrial company. The general potential of the methods was assessed to be very high. It was possible to elaborate prominent advantages and disadvantages of the approach.

Advantages of the described approach include assuring a consistency between the requirements themselves, as well as between product model and requirements. A quick response of the product layout for changing requirements may be achieved.

Disadvantages, at the moment, are the missing implementation of remaining functions of requirements management (assuring consistency and prioritization).

These disadvantages and the discussions themselves also lead to the formulation of further research and implementation activities (compare Section 9).

9. Summary and Outlook

The central objective of the research was to demonstrate an integration of model-based requirements management with a development process of gear systems, by means of applying graph-based design languages. For the innovative approach, two models were developed: one model for generating various gear system design variants for so-called “Electrical Multiple Units” (EMU), which are commonly applied in trams, suburban trains, or high-speed railway vehicles; and a second model for requirements management (RM), which includes state of the art RM-know-how.

The RM-Model as well as the model for product design variants have been implemented by means of graph-based design languages that are based on UML (Unified Modelling Language). In a graph-based design language, the engineering objects represent the vocabulary and the required model transformations represent the rules. In a so-called “production system”, the rules are executed in order to instantiate the vocabulary classes. This compilation process builds up a central data model. The model instances are displayed by a design graph. Within the research project, an engineering framework called “DesignCompiler43”, an eclipse-based software tool, has been used for compilation of the design language models.

The combination of both design languages enables a linkage of requirements to modules or components of the product, or to the product as a whole, regardless of the point at which the product life cycle is situated.

In addition to the current state, the approach is supposed to be expanded towards supporting the collection of requirements. A text-processing tool with word-analysis techniques can enable a fully automated transformation of textual requirements specification as a main input towards a composition of a class diagram based on specifications. This future process can lead to an instance graph of entered requirements. Further work is also planned in order to investigate analysis possibilities which are given by the polyhierarchical requirements model. Additionally, the implementation of remaining functions of requirements management (e.g., assuring consistency, . . .) is planned. In the future, the impact of agile and lean principles to model-based requirements engineering can also be a promising field for research.

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

Author Contributions: The research was carried out and the paper was written in a cooperative effort. Main research contributions concern the following points: Kevin Holder: descriptive study in the industrial company, generation of a graph-based model of a gear system; Andreas Zech: fundamentals of requirements engineering, generation of a general graph-based model of requirements engineering; Manuel Ramsaier: generation of graph-based models based on UML; Ralf Stetter: research methodology, fundamentals of requirements engineering; Hans-Peter Niedermeier: descriptive study in the industrial company, validation; Stephan Rudolph: fundamentals of graph-based languages; Markus Till: industrial application of graph-based languages, life-cycle issues.

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

References

- Bernard, R.; Irlinger, R. About watches and cars: Winning R&D strategies in two branches. In Proceedings of the Engineering Design—The Art of Building Networks, Garching, Germany, 4 April 2016.
- Hruschka, P. *Business Analysis und Requirements Engineering: Produkte und Prozesse Nachhaltig Verbessern*; Hanser: München, Germany, 2014.
- Ebert, C.; Jastram, M. ReqIF: Seamless Requirements Interchange Format between Business Partners. *IEEE Softw.* **2012**, *29*, 82–87. [[CrossRef](#)]
- Blessing, L.T.M.; Chakrabarti, A. *DRM, a Design Research Methodology*; Springer: London, UK, 2009.
- Kohn, A. Entwicklung einer Wissensbasis für die Arbeit mit Produktmodellen. Ph.D. Thesis, Lehrstuhl für Produktentwicklung der Technischen Universität München, Garching bei Muenchen, Germany, November 2013.
- Darlington, M.J.; Culley, S.J. A model of factors influencing the design requirement. *Des. Stud.* **2004**, *25*, 329–350. [[CrossRef](#)]
- Jiao, J.; Chen, C.H. Customer requirement management in product development: A review of research issues. *Concurr. Eng. Res. Appl.* **2006**, *14*, 173–185. [[CrossRef](#)]
- Zhang, Z.; Li, X.; Liz, Z. A closed-loop based framework for design requirement management. In Proceedings of the 21st ISPE Inc. International Conference on Concurrent Engineering, Beijing, China, 8–11 September 2014.
- ISO/IEC/IEEE 29148:2011: Systems and Software Engineering—Life Cycle Processes—Requirements Engineering. 2011. Available online: <http://ieeexplore.ieee.org/document/6146379/> (accessed on 29 August 2017).
- ISO/IEC 15288:2008 Systems and Software Engineering—System Life Cycle Processes. 2008. Available online: <https://www.iso.org/standard/43564.html> (accessed on 29 August 2017).
- ISO 9001:2015: Quality Management Systems—Requirements. 2015. Available online: <https://www.iso.org/standard/62085.html> (accessed on 29 August 2017).
- Morkos, B.; Mathieson, J.; Summers, J.D. Comparative analysis of requirements change prediction models: Manual, linguistic, and neural network. *Res. Eng. Des.* **2014**, *25*, 139–156. [[CrossRef](#)]

13. Mattmann, I.; Gramlich, S.; Kloberdanz, H. The malicious labyrinth of requirements—Three types of requirements for a systematic determination of product properties. In Proceedings of the 20th International Conference on Engineering Design (ICED15), Milan, Italy, 27–30 July 2015.
14. Becattini, N.; Cascini, G.; Rotini, F. Requirements checklists: Benchmarking the comprehensiveness of the design specification. In Proceedings of the 20th International Conference on Engineering Design (ICED15), Milan, Italy, 27–30 July 2015.
15. Chen, Z.Y.; Yao, S.; Lin, J.Q.; Zeng, Y.; Eberlein, A. Formalisation of product requirements: From natural language descriptions to formal specifications. *Int. J. Manuf. Res.* **2007**, *2*, 362–387. [[CrossRef](#)]
16. Bühne, S.; Herrmann, A. Handbuch Requirements Management nach IREB Standard. Aus- und Weiterbildung zum IREB Certified Professional for Requirements Engineering Advanced Level “Requirements Management”. IREB e.V.: 2015. Available online: https://www.ireb.org/content/downloads/14-handbook-cpre-advanced-level-requirements-modeling/ireb_cp_re_handbuch_requirements_modeling_advanced_level_v1.3.pdf (accessed on 29 August 2017).
17. Griffin, A.; Hauser, J. The Voice of the Customer. *Mark. Sci.* **1993**, *12*, 1–27. [[CrossRef](#)]
18. Sharif Ullah, A.M.M.; Tamaki, J. Analysis of Kano-Model-Based Customer Needs for Product Development. *Syst. Eng.* **2011**, *14*, 154–172. [[CrossRef](#)]
19. Sharif Ullah, A.M.M.; Sato, M.; Watanabe, M.; Mamunur Rashid, M. Analysis of Kano-Model-Based Customer Needs for Product Development. *Int. J. Autom. Technol.* **2016**, *10*, 132–143. [[CrossRef](#)]
20. Nambisan, S. Designing Virtual Customer Environments for New Product Development: Toward a Theory. *Acad. Manag. Rev.* **2002**, *27*, 392–413.
21. Vezzetti, E.; Alemanni, M.; Morelli, B. New product development (NPD) of ‘family business’ dealing in the luxury industry: Evaluating maturity stage for implementing a PLM solution. *Int. J. Fash. Des. Technol. Educ.* **2016**, *10*, 219–229. [[CrossRef](#)]
22. Salado, A.; Nilchiani, R. A Categorization Model of Requirements Based on Max-Neef’s Model of Human Needs. *Syst. Eng.* **2014**, *17*, 348–360. [[CrossRef](#)]
23. Martin, S.; Aurum, A.; Jeffery, R.; Paech, B. Requirements Engineering Process Models in Practice. In Proceedings of the Seventh Australian Workshop on Requirements Engineering, AWRE 2002, Melbourne, Australia, 2–3 December 2002.
24. Batra, M.; Bhatnagar, A. A Comparative Study of Requirements Engineering Process Model. *Int. J. Adv. Res. Comput. Sci.* **2017**, *8*, 740–745.
25. Almfelt, L.; Berglund, F.; Nilsson, P.; Malmqvist, J. Requirements management in practice: Findings from an empirical study in the automotive industry. *Res. Eng. Des.* **2006**, *17*, 113. [[CrossRef](#)]
26. Ramesh, B.; Jarke, M. Towards Reference Models for Requirements Traceability. *IEEE Trans. Softw. Eng.* **2001**, *27*, 58–93. [[CrossRef](#)]
27. Siemens: Teamcenter Systems Engineering. Available online: <https://www.plm.automation.siemens.com/de/products/teamcenter/systems-engineering-software/> (accessed on 29 August 2017).
28. IBM: Rational DOORS. Available online: <http://www-03.ibm.com/software/products/de/ratidoor> (accessed on 29 August 2017).
29. Serena: Serena Dimensions RM. Available online: <http://www.serena.com/index.php/de/products/more-products/dimensions-rm/> (accessed on 29 August 2017).
30. Carrillo de Gea, J.M.; Nicolas, J.; Fernandez Aleman, J.L.; Toval, A.; Ebert, C.; Vizca, A. Requirements engineering tools: Capabilities, survey and assessment. *Inf. Softw. Technol.* **2012**, *54*, 1142–1157. [[CrossRef](#)]
31. Object Management Group: Requirements Interchange Format (ReqIF) Version 1.2. Available online: <http://www.omg.org/spec/ReqIF/1.2/PDF> (accessed on 29 August 2017).
32. Eclipse: RMF/ProR. Available online: <http://www.eclipse.org/rmf/> (accessed on 29 August 2017).
33. Störrle, H.; Kucharek, M. The requirements editor RED. In Proceedings of the European Conference on Modelling Foundations and Applications (ECMFA 2013), Montpellier, France, 1–5 July 2014.
34. DTU: RED. The Requirements Engineering Workplace. Available online: <http://www2.compute.dtu.dk/~hsto/RED/index.html> (accessed on 29 August 2017).
35. Sudin, M.N.; Ahmed-Kristensen, S.; Andreasen, M.M. The role of a specification in the design process: A case study. In Proceedings of the International Design Conference, Cavtat-Dubrovnik, Croatia, 17–20 May 2010.
36. Li, X.; Ahmed-Kristensen, S. Understand the design requirement in companies. In Proceedings of the 20th International Conference on Engineering Design (ICED15), Milan, Italy, 27–30 July 2015.

37. Borgianni, Y.; Rotini, F. Stakeholders' diverging perceptions of product requirements: Implications in the design practice. In Proceedings of the 20th International Conference on Engineering Design (ICED15), Milan, Italy, 27–30 July 2015.
38. Finger, S.; Rinderle, J. *A Transformational Approach to Mechanical Design Using a Bond Graph Grammar*; Technical Report; Engineering Design Center of Carnegie Mellon University: Pittsburgh, PA, USA, January 1990.
39. Le Duigou, J.; Bernard, A. Product Lifecycle Management Model for Design Information Management in Mechanical Field. In Proceedings of the 21st CIRP Design Conference, Daejeon, Korea, 27–29 March 2011; pp. 207–213.
40. Eckert, C.; Albers, A.; Bursac, N.; Chen, H.X.; Clarkson, P.J.; Gericke, K.; Gladysz, B.; Maier, J.F.; Rachenkova, G.; Shapiro, D.; et al. Integrated Product and Process Models: Towards an Integrated Framework and Review. In Proceedings of the 20th International Conference on Engineering Design (ICED15), Milan, Italy, 27–30 July 2015.
41. Rudolph, S. Aufbau und Einsatz von Entwurfssprachen für den Ingenieurentwurf. In *Forum Knowledge Based Engineering*; CAT-PRO: Stuttgart, Germany, 2003.
42. Arnold, P.; Rudolph, S. Bridging the Gap between Product Design and Product Manufacturing by Means of Graph-Based Design Languages. In Proceedings of the TMCE, Karlsruhe, Germany, 7–11 May 2012.
43. Groß, J.; Rudolph, S. Generating simulation models from UML—A FireSat example. In Proceedings of the 2012 Symposium on Theory of Modeling and Simulation—DEVS Integrative M&S Symposium, Orlando, FL, USA, 26–29 March 2012.
44. Rudolph, S. On the problem of multi-disciplinary system design—And a solution approach using graph-based design languages. In Proceedings of the 1st ACCM Workshop on Mechatronic Design, Linz, Austria, 30 November 2012.
45. Eisenbahn-Kurier. Available online: <http://www.eisenbahn-kurier.de/startbeitraege/126-startseite/1518-bombardier-veranstaltet-erstes-internationales-forum-fuer-betreiber-von-flexx-drehgestellen> (accessed on 29 August 2017).
46. Ammenwerth, E. Die Modellierung von Anforderungen an die Informationsverarbeitung im Krankenhaus. Ph.D. Thesis, Medizinische Fakultät Heidelberg der Ruprecht-Karls-Universität, Heidelberg, Germany, May 2000.
47. Martin, J. *Information Engineering. Book II: Planning & Analysis*; Prentice Hall: Englewood Cliffs, NJ, USA, 1989.



© 2017 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).