EFFICIENT FORMALISATION OF TECHNICAL REQUIREMENTS FOR GENERATIVE ENGINEERING

Gräßler, Iris (1);
Preuß, Daniel (1);
Brandt, Lukas (2);
Mohr, Michael (3)

1: Heinx Nixdorf Institute / Paderborn University;
2: Atos Information Technology GmbH;
3: EDAG Engineering GmbH

ABSTRACT

Currently, engineers need to manually analyse requirement specifications for determining parameters to create geometries in generative engineering. This analysis is time-consuming, error-prone and causes high costs. Generative engineering tools (e.g. Synera) cannot interpret natural language requirements directly. The requirements need to be formalised in a machine-readable format. AI algorithms have the potential to automatically transform natural language requirements into such a formal, machine-readable representation. In this work, a method for formalising requirements for generative engineering is developed and implemented as a prototype in Python. The method is validated in a case example using three products of an automotive engineering service provider. Requirements to be formalised are identified in the specifications of these three products, which are used as a test set to evaluate the performance of the method. The results show that requirements for generative engineering are formalised with high performance (F1 of 86.55 %). By applying the method, efforts and therefore costs for manually analysing requirements regarding parameters for generative engineering are reduced.

Keywords: Requirements, Artificial intelligence, Semantic data processing, Computational design methods

Contact:
Preuß, Daniel
Heinx Nixdorf Institute / Paderborn University
Germany
daniel.preuss@hni.upb.de

Cite this article: Gräßler, I., Preuß, D., Brandt, L., Mohr, M. (2023) 'Efficient Formalisation of Technical Requirements for Generative Engineering', in *Proceedings of the International Conference on Engineering Design (ICED23),* Bordeaux, France, 24-28 July 2023. DOI:10.1017/pds.2023.160

1 INTRODUCTION

Generative engineering of components requires multiple parameters to be considered for creating geometries. These parameters are for example available design space, materials or the manufacturing process to. Generative engineering is part of the automated engineering process and is carried out in tools like Synera (Synera GmbH, 2022). Currently, the required parameters have to be determined manually by engineers through analysing requirement documents and modelling them as restrictions in the generative engineering tool. This analysis is time-consuming, error-prone and causes high costs (Ambriola and Gervasi, 2006). This is especially challenging in the engineering of components for complex technical systems, because requirement sets can contain several hundred requirements (Fernandes et al., 2015; Gräßler et al., 2018; Gräßler et al., 2016). Due to the high amount of requirements, there is a need for an automated analysis. When engineering components of complex technical systems, 80 % of the requirements are recorded in natural language (Mich et al., 2004; Neill and Laplante, 2003). Natural language is expressive and used for communication between a wide range of stakeholders (Casamayor et al., 2010; Gräßler, 2017). However, natural language is inherently ambiguous (Ceccato et al., 2004). Even more, the use of natural language in requirements makes them unclear (Wilson et al., 1997). It is not possible for generative engineering tools to directly interpret natural language requirements (Meth et al., 2015). Therefore, there is a need for an automated formalisation of natural language requirements so that they can be interpreted by generative engineering tools. Formalisation means extracting unstructured information embedded in texts and formatting it in a structured and machine-readable data format (Jurafsky and Martin, 2021).

In software engineering, algorithms based on Artificial Intelligence (AI) are applied to natural language processing for requirements engineering. AI algorithms have the potential to automatically transform natural language requirements into a formal, unambiguous representation (Giannakopoulou et al., 2021). These approaches can be adapted to the context of requirements in generative engineering of components. Based on the characterisation of parameters defined in the requirements, suitable approaches for formalisation must be selected and appropriate models trained. For example, it needs to be evaluated whether Named-Entity-Recognition is applicable for extracting the prescribed manufacturing process of the component. In addition, the performance of the approaches in the formalisation of requirements must be assessed. Only approaches with high performance are applicable to the problem. This research therefore aims at answering the following two research questions (RQ):

- *RQ 1: "Which approaches are applicable to formalise requirements in generative engineering of components for complex technical systems?" and*
- *RQ 2: "How do approaches for formalising requirements in generative engineering of components perform in complex technical systems?"*

This paper is divided into seven sections. The introduction (cf. Section 1) is followed by the methodology chosen to answer the research questions (cf. Section 2). Subsequently, an analysis to identify needs of formalising requirements for generative engineering and to derive success criteria of a method for formalising requirements is carried out (cf. Section 3). Then, approaches for formalising requirements are identified (cf. Section 4). Based on the success criteria and the identified approaches, a method for formalising requirements in generative engineering of components is developed (cf. Section 5). The performance of the developed method is evaluated (cf. Section 6). Finally, a summary of the results and an outlook on further research perspectives is given (cf. Section 7).

2 METHODOLOGY

This paper is part of the Federal Ministry for Economic Affairs and Climate Action funded research project BIKINI: bionics and artificial intelligence for sustainable integration in product development for resource efficient lightweight design (03LB3018C, https://bikini-projekt.de/). The research follows the approach for application-bound sciences according to Ulrich (Ulrich, 1982). The approach by Ulrich is chosen due to its application-orientation and the integrative consideration of engineering and management sciences. The approach is depicted in Figure 1.

The first step is to identify problems of practical relevance. The scientific need and research questions of the contribution are derived. Success criteria of formalising requirements for generative engineering are elicited via literature study and interviews with practitioners. Two interviews are conducted, each with a requirements engineer from the engineering service providers EDAG Engineering GmbH

https://doi.org/10.1017/pds.2023.160 Published online by Cambridge University Press

(short: EDAG). As a second step, a literature survey according to Machi and McEvoy (Machi and McEvoy, 2012) is carried out to specify problem-relevant procedures. Thus, existing approaches for the formalisation of natural language are identified. Among those, relevant ones are selected. Based on the success criteria and the selected approaches, a method for formalising requirements for generative engineering is developed in the third step. This method is implemented as a prototype in Python. Requirements data are prepared (labelling of parameters and formatting the data) to develop the method and to form a training set for models for formalising requirements. For this purpose, requirement specifications from three past development projects of EDAG are used. The method is validated in a case example using the products "assembly latch hood", "adjustable stopper hood" and "trunk curtain roller blind". 76 requirements to be formalised are identified in these three specifications of EDAG, which are used as a test set to evaluate the performance of the method. In addition, an interview is conducted with the same two requirements engineers from EDAG to discuss the implemented method and results. The results are presented to them and they evaluate whether the method adds value to generative engineering in practice.

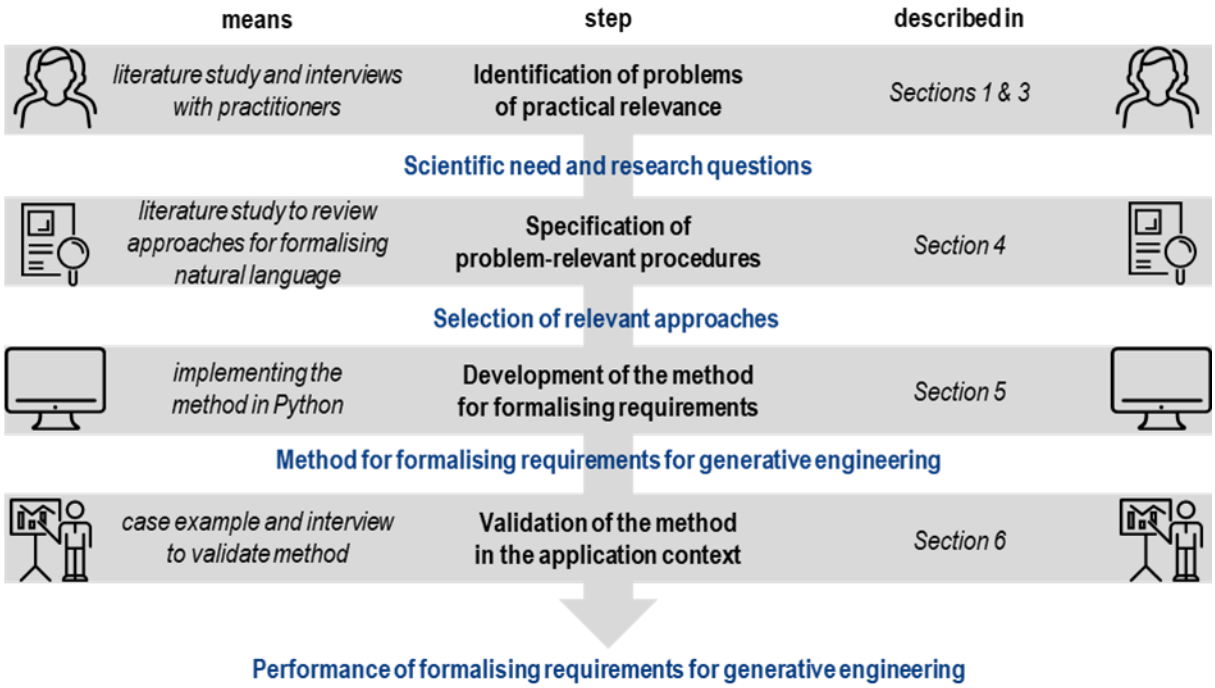

Figure 1. Methodology for application bound sciences based on (Ulrich, 1982)

3 GENERATIVE ENGINEERING AND SUCCESS CRITERIA

In the following, generative engineering is described and success criteria of the method for formalising requirements for generative engineering are determined (cf. Section 2). For this, two semi-structured interviews (each 30 minutes) are performed with two requirements engineers from EDAG to capture needs from industrial practice.

Generative engineering: In generative engineering, geometries are generated automatically based on restrictions. The restrictions are derived from requirements that are contained in requirement specifications. Generated geometries are optimised with regard to certain criteria. For example, a door module can be optimised with regard to the criterion "mass" and at the same time comply with restrictions regarding different load cases. In generative engineering the goal is to define a workflow, which generates the component automatically, based on specific parameters. Since the process of modelling a component manually is very time-consuming, the generative approach saves time through providing a high level of automation (Gräßler, 2003). Software tools like Synera enable such an approach. An exemplary view of a Synera workflow for generating geometries (in this case for a door module) is shown in Figure 2.

https://doi.org/10.1017/pds.2023.160 Published online by Cambridge University Press

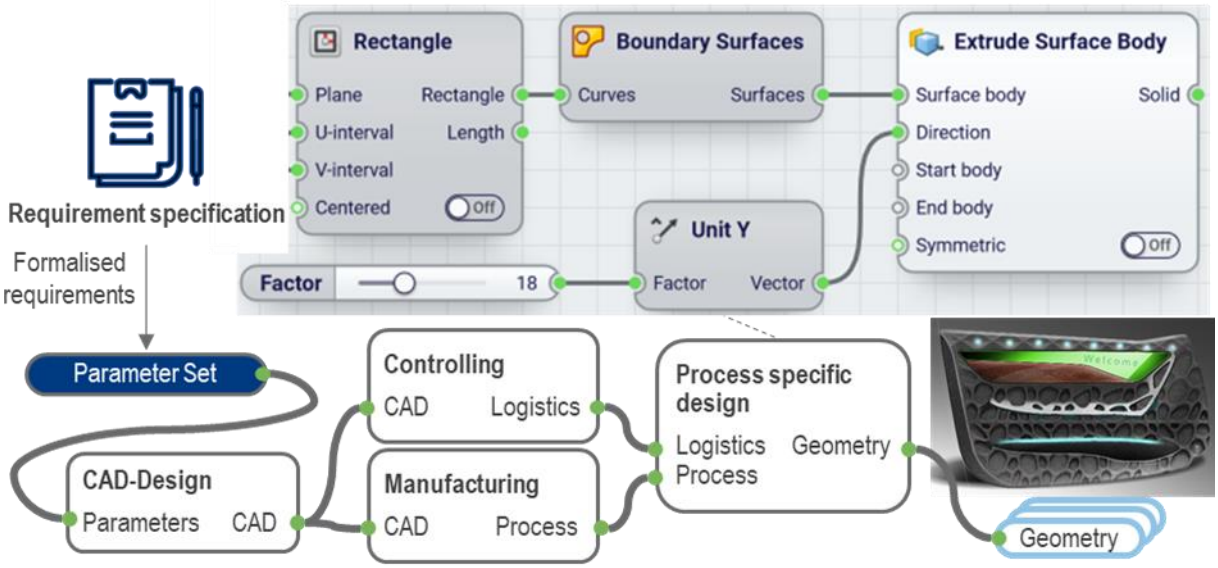

Figure 2. Generative engineering of a door module in Synera

4 STATE OF RESEARCH

A literature review is conducted to determine the state of research in natural language formalisation (cf. Section 4.1) and approaches that formalise natural language requirements (cf. Section 4.2). The review is performed to be able to select relevant approaches, to distinguish the approach from current research and to show the degree of novelty.

4.1 Formalising natural language

For formalising natural language text, Jurafsky and Martin divide approaches into three types of tasks (Jurafsky and Martin, 2021). Relation Extraction means recognising entities and relations among entities, such that a graph structure, e.g. a knowledge graph, can be constructed. The graph construction can either be executed from scratch, or alternatively the found entities can be linked with existing nodes to extend the graph. Event Extraction is described as identifying mentions of events in text, including recognising participating entities. Additionally, the temporal expression of the particular point, or interval, in time of the event has to be extracted as well as normalised in a structured format. In Template filling, a template is defined as a set of semantic keys and corresponding data types. Each template key has to be filled with concepts like time intervals, amounts, or ontology entities of the source text, which have the correct data type and are as a value compliant with the expected semantic of the key. A key with an assigned value is referred to as key-value pair (Du et al., 2021).

In the formalisation of requirements, events are negligible but knowledge graph construction through relation extraction and template filling are applicable approaches. Knowledge graphs are a very comprehensive formalisation of text, in which as much information as possible is contained, while key-value pairs are a very narrow but concrete formalisation. The selection of a text formalisation approach depends on the question how well-known the future questions are that will be answered via the extracted and formalised information. Both tasks have in common that an approach for entity recognition is necessary, such that the entities can either be analysed with regards to relationships in graphs or as possible values for keys. In this context, entities are words or phrases from the source text, and they belong to a named category, e.g. person, location, or organisation. Therefore, these entities are also called by the term Named-Entities and the recognition is called Named-Entity-Recognition (NER) (Jurafsky and Martin, 2021). For example, NER models enable detecting the phrase “Robert Bosch GmbH” in a text and automatically classify it as "supplier". Common algorithms for training a NER model based with appropriate training data are for example (Jurafsky and Martin, 2021):

- Hidden Markov Models
- Conditional Random Fields
- Recurrent Neural Networks
- Transformer

https://doi.org/10.1017/pds.2023.160 Published online by Cambridge University Press

A widely adopted benchmark for NER models is the CoNLL 2003 NER Task, in which persons, locations, organisations, and miscellaneous entities must be extracted from news reports (Sang, Tjong and Meulder, 2003). Currently, the best benchmark results from an approach using transformer-based embeddings (Wang et al., 2021).

4.2 Approaches for formalising natural language requirements

In the area of formalisation of requirements, the construction of entities and dependency-based structure, e.g. a graph, representing the natural language source text is often used as an intermediate step. The result of such graph transformation approaches is a graph that represents the object-oriented model of components of the target domain (Ilieva and Ormandjieva, 2005) or an entity relationship graph for a domain data model (Kashmira and Sumathipala, 2018). Another approach is chosen in the ARSENAL tool, which uses a graph with relationships and dependencies between entities to create a representation of requirements using Linear Temporal Logic (LTL) (Ghosh et al., 2014). The goal of other approaches is likewise to transfer natural language requirements into a logical formalism, e.g. a hierarchical clustering used to structure recognised entities and relationships/dependencies (Giantamidis et al., 2021) or a rule-based approach based on the found entities and grammatical dependencies (Koscinski et al., 2021). In another approach, product ideas (e.g. "Rolling toy as a point-of-sale device") are generated by AI using keywords, which are input manually (Zhu and Luo, 2023). AI constraint networks are used in SPARK to enable engineering despite incomplete information (Young et al., 1991).
In summary, the analysis of related work shows that there are no approaches for formalising requirements in the context of generative engineering of components of complex technical systems. So far, the performance of formalising approaches have not been assessed for this research area.

5 METHOD FOR FORMALISING REQUIREMENTS

Success criteria of the method: Resulting from literature study (cf. Section 1) and interviews, a total of eight success criteria are elicitated (cf. Table 1). Success criteria form requirements of the method: the term "success criteria" is used instead of "requirements" to improve readability, as the objects of research are also requirements. The success criteria are categorised as related to "input", “model capabilities", "model use" and "output" (Hamraz et al., 2013). The support needs to be able to process requirements in natural language. The requirements shall not need to follow a fixed template (e.g. templates like (Joppich et al., 2016)) and they must be in an industry-standard format (SC 1). The language of the specification needs to be English (SC 2). Relevant restrictions of generative engineering need to be formalised (SC 3). These restrictions were collected during the interviews. The requirements engineers noted that for generative engineering it should be possible to assess the sustainability of the component. For this purpose, additional parameters that enable a sustainability assessment need to be extracted (SC 4). Since results from interviews and literature study (Fernandes et al., 2015; Mich et al., 2004; Neill and Laplante, 2003) indicate that extracting parameters from requirements requires high effort, parameters need to be extracted automatically (SC 5). Information needed to apply the support have to be available in an industry-standard manner (e.g. only a realistic amount of training data is required) to be usable in practice was noted in the interviews (SC 6). The requirements engineers have expressed that the accuracy of formalisation must be high (SC 7) and that a common interchange format (SC 8) must be chosen for allowing import into generative engineering tools with low manual effort. Results of the literature review indicate that high accuracy in formalising requirements is hindered by the ambiguity of natural language (Casamayor et al., 2010).

Table 1. Success criteria for formalising requirements

ID	Category	Success Criteria
SC 1	Input	Process natural language requirements
SC 2	Input	Process English requirements
SC 3	Model capabilities	Formalise restrictions
SC 4	Model capabilities	Formalise criteria for sustainability assessment
SC 5	Model use	Reduce effort through automation
SC 6	Model use	Information in industry standard manner
SC 7	Output	Sufficient accuracy
SC 8	Output	Common interchange format

https://doi.org/10.1017/pds.2023.160 Published online by Cambridge University Press

The *method for formalising requirements for generative engineering* is developed based on the success criteria and the findings from analysing state of research. In generative engineering, various parameters must be extracted from the requirements. A narrow formalisation is needed so that the extracted information is machine-readable and can be interpreted using generative engineering tools. For this reason, a template filling approach in combination with NER is chosen (cf. Section 4). Transformer models are applied for this task, because they show best results for NER (Wang et al., 2021). The developed method is a combination of data-based AI and a rule-based approach.

To formalise requirements, key-value pairs are recognised in natural language requirements. An example of a key-value pair is visualised in Figure 3. The key-value pair of the requirement "The lifespan of the system must be at least 300.000 km." must be extracted. In this example, the key is "lifespan_range". The lifespan is specified as a unit of distance. The value of the key is "at least 300.000 km". Thus the key-value pair is "lifespan_range: ≥ 300.000 km". Various structure rules and text indicators are used to detect the key-value pair, which are defined in the domain lexicon. Value classes are defined per key-value pair. In this example, the value is a number (in combination with the relational operator "at least", or "≥" and the unit kilometer, or "km"). Values do not necessarily have to be real numbers in the context of formalising requirements. In the example requirement "The door module is to be manufactured using additive manufacturing", the key is "manufacturing process" and the value is "additive manufacturing".

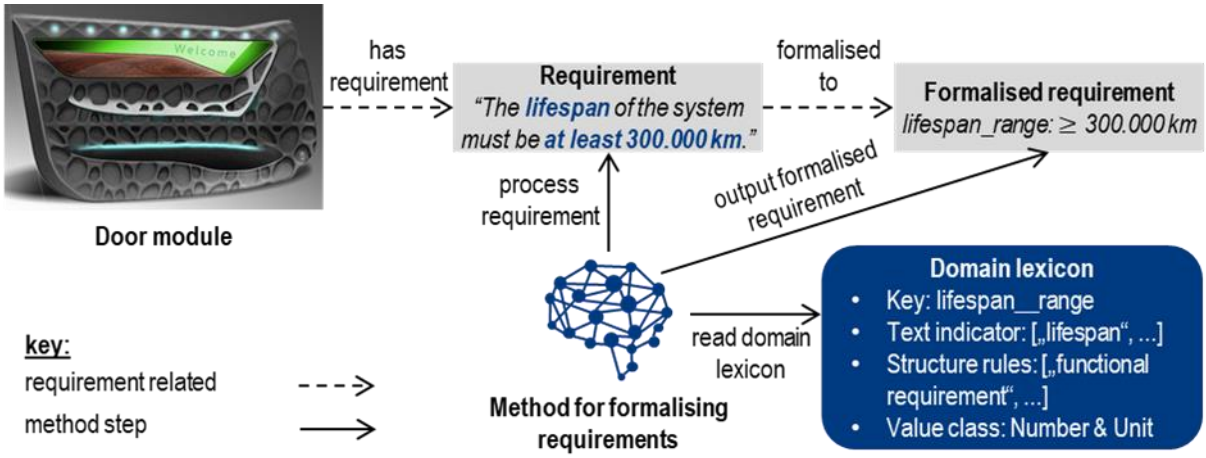

Figure 3. Formalising a requirement of a door module (example)

The method for formalising requirements consists of three steps. *Step 1:* The requirement candidates must be identified (cf. Figure 4). To reduce the runtime and extracting false positive results of the algorithm for formalising requirements, only certain requirements should be checked for containing certain key-value pairs. A requirement is only examined for a key-value pair if all structure rules are fulfilled. Requirements that meet these conditions are called requirement candidates. Structure rules are defined per key-value pair. These structure rules are used as a positive or negative list. For example, structure rules for the requirement regarding the lifespan of the door module are:

- Structure rule 1: "The key "lifespan_range" can only occur in a requirement if a unit of distance is present". (positive)
- Structure rule 2: "The key "lifespan_range" cannot occur in a functional requirement." (negative)

If a unit of distance (e.g. "km") is not present in a requirement, then the requirement is not checked whether a key-value pair exists for the key "lifespan_range" (structure rule 1). In addition, meta-data of the requirements can be used to form structure rules. A key-value pair of the key "lifespan_range" cannot occur in a functional requirement, which is why functional requirements are not checked for this key-value pair (structure rule 2).

Step 2: The requirement candidates are filtered by comparing them with text indicators of keys. Text indicators for "lifespan_range" are e.g. "lifespan" or "durability". These text indicators are defined manually in the domain lexicon. The definition can be partially automated, by identifying synonyms from lexical databases like WordNet (Fellbaum, 2005). All words of the requirement candidates (after removing stop words) are used as candidates for key-value pairs. The key-value pair candidates are lemmatised and compared directly with the lemmas of text indicators and searched for matches.

https://doi.org/10.1017/pds.2023.160 Published online by Cambridge University Press

Step 3: If at least one match is found, the key-value candidates are checked whether they contain a value for the matching key. Depending on the key, different approaches for recognising values are used. For example, the key "lifespan_range" can take on the values 0 to n and a unit of distance (like "km" or "kilometer"). This class of values is called "number with unit". Regular Expressions (RegEx) are used for recognition. If there are multiple possible values for a key, which do not follow a fixed syntax like the value class "number with units", NER is used. For example, the key "manufacturing_site" requires a value of the location of the manufacturing site (e.g. country "Germany" or city "Paderborn"). Pre-trained models exist for some NER tasks. In this approach, spaCy´s NER models are used for value class "location". For other value classes like "material" or "manufacturing_process", NER models are specifically trained. If there are few possible values for a key, a string matching approach is used. A list of strings for this value is manually defined. The lemmatised strings are matched with the lemmatised value candidates. For example, the key "energy_type" requires a value of the type of energy needed by a vehicle (e.g. "diesel" or "electric"). If the key-value pair candidate does not contain a value, which matches the key it is discarded as a candidate for this key. If the requirement contains a single matching value, it is used as the value for the key-value pair. If it is not possible to determine a unique value, then all value candidates must be verified manually. The procedure is illustrated in Figure 4.

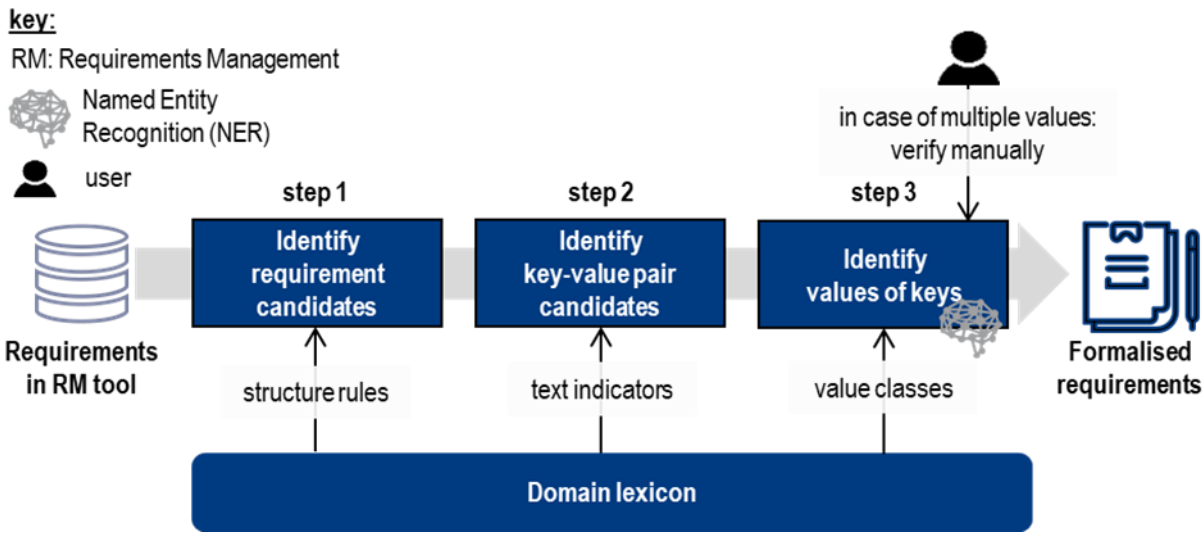

Figure 4. Procedure for formalising requirements

6 VALIDATION

To validate the method for formalising requirements, eight key-value pairs are identified and metrics are calculated within a case example to assess the performance of the method. The key-value pairs to be analysed were chosen so that all of the different formalisation approaches of the method can be tested. A semi-structured interview is conducted with two requirements engineers. The method and results of formalising requirements are presented to them and they evaluate the benefit for generative engineering when applying the method in practice.

Performance metrics

Requirements from three requirement specifications of EDAG ("assembly latch hood", "adjustable stopper hood" and "trunk curtain roller blind") are formalised to extract key-value pairs. The performance for extracting key-value pairs is measured. For this purpose, a test set of key-value pairs to be formalised is created. To create the test set, key-value pairs from the domain lexicon are labelled in the requirements of EDAG. The test set is used to assess the performance of the method using classifier metrics. In the labelled requirements of EDAG, there are eight different key-value pairs that have to be extracted from requirements. The key-value pairs are explained in Table 2.

https://doi.org/10.1017/pds.2023.160 Published online by Cambridge University Press

Table 2. Key-value pairs used in validation

Key	Value class	Explanation
manufacturing_process	NER model (own)	Manufacturing process to be used
manufacturing_site	NER model (spaCy)	Site to be manufactured in (location)
material	NER model (own)	Material to be used
energy_type	string matching	Energy type like fuel or electrical
mass	number with unit	Mass of the product
lifespan_range	number with unit	Lifespan of the product in unit of range
lifespan_time	NER model (own)	Lifespan of the product in unit of time
supplier_selected	NER model (spaCy)	Supplier of the product

own: trained NER model with EDAG data; spaCy – pre-trained NER model by spaCy

For measuring the performance in formalising requirements, the classifier metrics precision, recall and F1 are used (Alpaydın, 2019). The method is used to formalise 76 requirements from three specifications of EDAG extracting the key-value pairs described in Table 2. For each key, a set of requirements is selected, which contain values of the respective keys. The results are shown in Table 3.

Table 3. Results for formalising requirements

Key-value pair	Precision	Recall	F1	Num. req. for Test/Training
manufacturing_process	93.33 %	77.78 %	84.85 %	18/68
manufacturing_site	100.00 %	100.00 %	100.00 %	3/-
material	100.00 %	60.00 %	75.00 %	10/59
energy_type	100.00 %	100.00 %	100.00 %	4/-
mass	100.00 %	90.00 %	94.74 %	10/-
lifespan_range	100.00 %	100.00 %	100.00 %	10/-
lifespan_time	100.00 %	75.00 %	85.71 %	11/9
supplier_selected	100.00 %	77.78 %	87.50 %	9/-
Macro average	99.26 %	76.73 %	86.55 %	76/146

Num. req.: "Number of requirements"

Overall, the results show that the method extracts the key value pairs in the requirements with high performance (macro average F1 of 86.55 %). Especially the precision is high (99.26 %), which means that hardly any false positives have been detected. This indicates that the text indicators for detecting keys have been defined correctly. The "string matching" and "number with unit" approaches perform well (avg. F1 100 % and 97.97 %). The pre-trained spaCy models for extracting "manufacturing_site" and "supplier_selected" also perform well in this context (F1 of 100.00 % and 87.50 %).

Interview

A semi-structured interview was conducted with two requirements engineers from EDAG. The method and results were presented to the requirements engineers and questions (Q 1 to Q 3) were asked. The results are summarized by the engineers' responses (R 1 to R 3). Direct quotes are printed in italic.

Q 1: "Is the information sufficient to perform generative engineering? Is further information needed?"

R 1: The method is applicable to extracting relevant parameters for generative engineering including sustainability assessment. The formalisation approaches within the method (e.g. "number with unit") are transferable to other parameters, e.g. "volume of components". "Not all parameters can be read directly from the requirements. Some of them have to be derived". These parameters are material properties, e.g. waterproof. Due to the expressiveness of natural language, not all requirements can be parameterised.

Q 2: Is the method adding value to generative engineering?

R 2: Because of the amount of parameters needed for engineering a specific part in complex technical systems, extracting the parameters manually is time-consuming. Especially by integrating the formalisation into the workflow of generative engineering, time is saved.

Q 3: Is the method applicable? What information is needed to enable applicability in practice?

R 3: The requirements specification needed for the application of the method is available in our company as the basis for engineering." Internally, a process has to be defined for formalising the requirements, as

https://doi.org/10.1017/pds.2023.160 Published online by Cambridge University Press

the Synera engineer does not have access to the requirements specification. Since NER models only require a low amount of training data, effort for generating data in practice is acceptable.

Conclusion

The method supports formalising natural language requirements in English that do not follow a fixed template (SC 1 and SC 2 are fulfilled). Because of multiple approaches (like NER models or "number with unit"), the method is transferable to different types of parameters. Nevertheless, some parameters e.g. material properties, need to be manually derived from experts and cannot be formalised directly using this approach (SC 3 and SC 4 are partially fulfilled). The effort of formalising requirements is reduced. An implemented interface for generative engineering tool is required to further decrease effort (SC 5 is partially fulfilled). The information needed is available in practice and the effort for generating training data is low (SC 6 is fulfilled). In validation, only 146 requirements for training were needed to achieve the displayed high performance (F1 of 86.55 %, SC 7 is fulfilled). The extracted parameters are exportable in CSV format allowing import in generative engineering tools (SC 8 is fulfilled).

7 SUMMARY AND OUTLOOK

In this research, a method for formalising requirements for generative engineering of components for complex technical systems is developed. Within the method, the approaches "string matching", "number with unit" and NER models are used to formalise requirements for generative engineering (RQ 1). The validation is carried out via formalising requirements from three EDAG specifications and conducting an interview with two requirements engineers. Requirements of generative engineering are formalised with high performance (F1 of 86.55 %, RQ 2). By applying the method, effort of manually analysing requirements regarding parameters for generative engineering is reduced. The research presented contributes to the application of AI methods in engineering. Further research potential lies in using formalised requirements for dependency analysis (Gräßler et al., 2020; Gräßler and Yang, 2016) as well as importing formalised requirements into system modelling tools for impact analysis of requirement changes (Gräßler et al., 2022; Gräßler and Pöhler, 2020). Moreover, the potential of applying NLP approaches to automatic elicitation of requirements needs to be explored. This benefits engineers through creating candidate solutions of generative systems using fewer constraints and enabling an explorative approach. The approach needs to be tested with a higher number of requirements from different companies to assess effects of different requirement representations on the results.

REFERENCES

Alpaydın, E. (2019), *Maschinelles Lernen*, De Gruyter Studium, Vol. 2, De Gruyter Oldenbourg, Berlin.

Ambriola, V. and Gervasi, V. (2006), “On the Systematic Analysis of Natural Language Requirements with C IRCE”, Automated Software Engineering, Vol. 13 No. 1. https://doi.org/10.1007/s10515-006-5468-2

Casamayor, A., Godoy, D. and Campo, M. (2010), “Identification of non-functional requirements in textual specifications”, *Information and Software Technology*, Vol. 52 No. 4, pp. 436–445. https://doi.org/10.1016/j.infsof.2009.10.010

Ceccato, M., Kiyavitskaya, N., Zeni, N., Mich, L. and Berry, D. M. (2004), *Ambiguity Identification and Measurement in Natural Language Texts*, Technical report.

Du, X., Rush, A. M. and Cardie, C. (2021), “Template Filling with Generative Transformers”, *Proceedings of the North American Chapter of the Association for Computational Linguistics*, pp. 909–914. http://dx.doi.org/10.18653/v1/2021.naacl-main.70

Synera GmbH (2022), “Synera”, available at: https://www.synera.io/ (accessed 13 November 2022).

Fellbaum, C. (2005), "WordNet and wordnets", *Encyclopedia of Language and Linguistics*, No. 2, pp. 665-670.

Fernandes, J., Henriques, E., Silva, A. and Moss, M.A. (2015), “Requirements change in complex technical systems. An empirical study of root causes”, *Research in Engineering Design*, Vol. 26 No. 1, pp. 37–55. http://dx.doi.org/10.1007/s00163-014-0183-7

Giannakopoulou, D., Pressburger, T., Mavridou, A. and Schumann, J. (2021), “Automated formalization of structured natural language requirements”, *Information and Software Technology*, Vol. 137 No. 2. https://doi.org/10.1016/j.infsof.2021.106590

Giantamidis, G., Papanikolaou, G., Miranda, M., Salinas-Hernando, G., Valverde-Alcala, J., Veluru, S., and Basagiannis, S. (2021), “ReForm: A Tool for Rapid Requirements Formalization”, *Electronic Communications of the EASST*, Vol. 79 No. 0. http://dx.doi.org/10.14279/tuj.eceasst.79.1117

Ghosh, S., Elenius, D., Li, W., Lincoln, P., Shankar, N. and Steiner, W. (2014), ARSENAL: *Automatic Requirements Specification Extraction from Natural Language*. https://doi.org/10.48550/arXiv.1403.3142

https://doi.org/10.1017/pds.2023.160 Published online by Cambridge University Press

Gräßler, I. (2003), “Impacts of Information Management on Customized Vehicles and After Sales Services”, *International Journal of Computer Integrated Manufacturing*, Vol. 16 No. 7-8, pp. 566–570. https://doi.org/10.1080/0951192031000115714
Gräßler, I. (2017), “A new V-Model for interdisciplinary product engineering”, *Engineering for a changing world*, Ilmenau Scientific Colloquium, 11.09.-15.09., Ilmenau, Germany.
Gräßler, I., Dattner, M., Bothen, M. and Bothen, M. (2018), “Main Feature List as core success criteria of organizing Requirements Elicitation”, *R&D Management Conference*, 30.06.-04.07., Milan, Italy. http://dx.doi.org/10.31224/osf.io/grfcn
Gräßler, I., Hentze, J. and Yang, X. (2016), “Eleven Potentials for Mechatronic V-Model”, *6th International Conference Production Engineering and Management*, 29.09.-30.09., Lemgo, Germany.
Gräßler, I., Oleff, C. and Scholle, P. (2020), “Method for Systematic Assessment of Requirement Change Risk in Industrial Practice”, *Applied Sciences*, Vol. 10 No. 23, p. 8697. https://doi.org/10.3390/app10238697
Gräßler, I. and Pöhler, A. (2020), “Produktentstehung im Zeitalter von Industrie 4.0”, *Handbuch Gestaltung digitaler und vernetzter Arbeitswelten*, Springer Berlin Heidelberg, pp. 383–403. https://doi.org/10.1007/978-3-662-52979-9_23
Gräßler, I., Wiechel, D., Koch, A.-S., Preuß, D. and Oleff, C. (2022), “Model-based effect-chain analysis for complex systems”, *Proceedings of the Design Society: 17th International Design Conference*, 23-26.05.2022, Cavtat, Croatia, Cambridge University Press, Cambridge, UK. https://doi.org/10.1017/pds.2022.191
Gräßler, I. and Yang, X. (2016), “Interdisciplinary Development of Production Systems Using Systems Engineering”, *Procedia CIRP*, Vol. 50, pp. 653–658. http://dx.doi.org/10.1016/j.procir.2016.05.008
Hamraz, B., Caldwell, N., Wynn, D. and Clarkson, P. (2013), “Requirements-based development of an improved engineering change management method”, *Journal of Engineering Design*, Vol. 24 No. 11, pp. 765–793. http://dx.doi.org/10.1080/09544828.2013.834039
Ilieva, M.G. and Ormandjieva, O. (2005), “Automatic Transition of Natural Language Software Requirements Specification into Formal Presentation”, Springer, Berlin, Heidelberg, pp. 392–397. https://doi.org/10.1007/11428817_45
Joppich, R., Pflüger, A., Queins, S., Rupp, C., Schöne, K., Stuy, A. and Vöge, A. (2016), "Schablonen für alle Fälle", 3rd ed., SOPHIST GmbH.
Jurafsky, D. and Martin, J.H. (2021), *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*, Prentice-Hall International, NJ, London.
Kashmira, P. G. T. H. and Sumathipala, S. (2018), “Generating Entity Relationship Diagram from Requirement Specification based on NLP”, *3rd International Conference on Information Technology Research (ICITR)*. http://dx.doi.org/10.1109/ICITR.2018.8736146
Koscinski, V., Gambardella, C., Gerstner, E., Zappavigna, M., Cassetti, J. and Mirakhorli, M. (2021), “A Natural Language Processing Technique for Formalization of Systems Requirement Specifications”, *29th IEEE International Requirements Engineering Conference workshops*, IEEE, Piscataway, NJ, pp. 350–356. https://doi.org/10.1109/REW53955.2021.00062
Machi, L.A. and McEvoy, B.T. (2012), *The literature review: Six steps to success*, 2. ed., Corwin.
Meth, H., Mueller, B. and Maedche, A. (2015), “Designing a Requirement Mining System”, *Journal of the Association for Information Systems*, Vol. 16 No. 9, pp. 799–837. http://dx.doi.org/10.17705/1jais.00408
Mich, L., Franch M. and Inverardi, P. N. (2004), “Market research for requirements analysis using linguistic tools”, *Requirements Engineering*, Vol. 9 No. 1, pp. 40–56. https://doi.org/10.1007/s00766-003-0179-8
Neill, C.J. and Laplante, P.A. (2003), “Requirements engineering: The state of the practice”, *IEEE Software*, Vol. 20 No. 6, pp. 40–45. http://dx.doi.org/10.1109/MS.2003.1241365
Sang, E. F. Tjong, K. and de Meulder, F. (2003), “Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition”, *Proceedings of CoNLL*. https://doi.org/10.48550/arXiv.cs/0306050
Ulrich, H. (1982), “Anwendungsorientierte Wissenschaft”, *Die Unternehmung*, Vol. 36, pp. 1–10.
Wang, X., Jiang, Y., Bach, N., Wang, T., Huang, Z., Huang, F. and Tu, K. (2021), “Automated Concatenation of Embeddings for Structured Prediction”, Association for Computational Linguistics, USA, pp. 2643–2660. https://doi.org/10.48550/arXiv.2010.05006
Wilson, W.M., Rosenberg, L.H. and Hyatt, L.E. (1997), “Automated analysis of requirement specifications”, ICSE 97: *19th Annual Conference on Software Engineering*, Boston, Massachusetts, United States, Association for Computing Machinery, New York, pp. 161–171. https://doi.org/10.1145/253228.253258
Young, R. E., Greef, A., and O'Grady, P. (1991), “SPARK: an artificial intelligence constraint network system for concurrent engineering“, *Artificial intelligence in design ´91*, Butterworth-Heinemann, pp. 79–94. https://doi.org/10.1016/B978-0-7506-1188-6.50008-9
Zhu, Q. and Luo, J. (2023), "Generative Design Ideation: A Natural Language Generation Approach", *Design Computing and Cognition’22*, Springer International Publishing, pp. 39–50. https://doi.org/10.48550/arXiv.2204.09658