Design Science

Use case identification of natural language system requirements with graph-based clustering

Simon Schleifer [1], Adriana Lungu[2], Benjamin Kruse[2], Sebastiaan van Putten[2], Stefan Goetz [1] and Sandro Wartzack [1]

[1]*Engineering Design, Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, Germany*
[2]*Technical Development, AUDI AG, Ingolstadt, Germany*

Abstract

Due to the ever-increasing complexity of technical products, the quantity of system requirements, which are typically expressed in natural language, is inevitably rising. Model-based formalization through the application of Model-based Systems Engineering is a common solution to cope with rising complexity. Thereby, grouping requirements to use cases forms the first step towards model-based requirements and allows to improve the understanding of the system. To support this manual and subjective task, automation by artificial intelligence and methods of natural language processing are needed. This contribution proposes a novel pipeline to derive use cases from natural language requirements by considering incomplete manual mappings and the possibility that one requirement contributes to multiple use cases. The approach utilizes semi-supervised requirements graph generation with subsequent overlapping graph clustering. Each identified use case is described by keyphrases to increase accessibility for the user. Industrial requirement sets from the automotive industry are used to evaluate the pipeline in two application scenarios. The proposed pipeline overcomes limitations of prior work in the practical application, which is emphasized by critical discussions with experts from the industry. The proposed pipeline automatically generates proposals for use cases defined in the requirement set, forming the basis for use case diagrams.

Keywords: Model-based Systems Engineering (MBSE), Natural Language Processing (NLP), Overlapping Graph Clustering, Requirements Engineering, Use Case Identification

Received 10 May 2024
Revised 01 July 2025
Accepted 02 July 2025

Corresponding author
Simon Schleifer
schleifer@mfk.fau.de

Des. Sci., vol. 11, e30
journals.cambridge.org/dsj
DOI: 10.1017/dsj.2025.10019

1. Introduction

The automotive industry faces new challenges such as connected and autonomous vehicles (Pérez-Moure *et al.* 2024), which inevitably lead to an increased complexity in the development. A possible source of so-called system requirements is a System of Systems (SoS) in the domain of transportation (Wilking *et al.* 2024), which can comprise other road users, such as micro-mobility solutions like e-scooters, and charging infrastructure. Additionally, system requirements directly originate from stakeholders such as users, governments in the form of regulations (Pohl & Rupp 2021) and legacy projects, see Figure 1a.

The usage of natural language requirements is still the predominant solution to apprehend these requirements because of their accessible comprehension and their

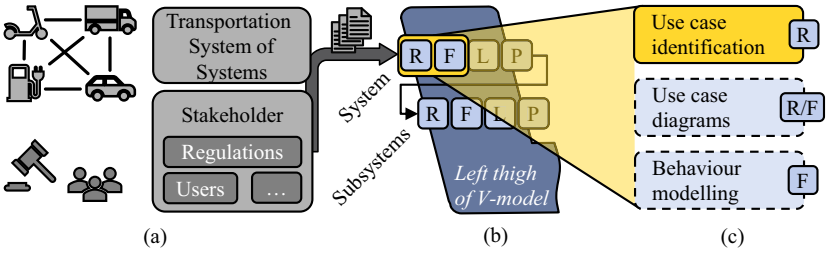

Figure 1. Sources of system requirements (a) as a basis for product development (b) highlighting the use case identification on the system level as the focus of this article (c).

flexibility (Pohl & Rupp 2021). In industrial practice, however, problems arise due to the ambiguity of natural language requirements. Even if the requirement formulation coincides with standardized templates, heterogeneous phrases are observed. In addition, multilingual requirements specifications are common in globally operating companies. These problems limit the applicability of natural language requirements. The use of formal models instead of textual requirements helps to avoid these problems. This is referred to as model-based requirements and characterized by more formal and consistent representations compared to natural language (Mordecai & Dori 2017). In addition, there are advantages through an improved understanding of requirements, their traceability, and a targeted system decomposition (Salado & Wach 2019).

Model-based Systems Engineering (MBSE) is one solution for the automotive industry to master the increasing complexity. As this industry traditionally worked with natural-language requirements, the utilization of model-based requirements often marks the starting point for consistent documentation and usage of the system requirements throughout the product development process.

The V-model structures the MBSE activities (Walden *et al.* 2015) by describing a holistic methodology for the product development of complex products. For this, the left thigh breaks down the system vertically into different levels comprising subsystems and system elements, see Figure 1b. The Requirements-Functions-Logical-Physical approach (RFLP) (Kleiner & Kramer 2013) is used for horizontal decomposition on each system level (Krog *et al.* 2022). This overall decomposition is followed by the implementation at the tip of the V-model. The right thigh then describes the integration (Verein Deutscher Ingenieure 2021). This article deals with the system level. The vehicle is an example for the focussed level in the context of an SoS, whereas the brake system forms a subsystem of the vehicle system.

Requirement engineers acquire new requirements specifications from stakeholder demands and legacy specifications, for example. Next, the system architect uses these requirements specifications to generate a formalized view of the system. The identification of use cases is usually the first step for systems architects when utilizing model-based requirements in the context of MBSE, see Figure 1c. This step creates a link between a large body of system requirements without a clear structure and system use cases (Walden *et al.* 2015). A use case, for example, 'unlock vehicle', is described by a set of natural language system requirements and is part of the requirement viewpoint of RFLP. Subsequent steps, such as the generation of use case diagrams and behaviour modelling define the transition

https://doi.org/10.1017/dsj.2025.10019 Published online by Cambridge University Press

to and, respectively, the functional viewpoint itself. The concept of use cases is important, as it establishes an unambiguous understanding of the system functionalities from a user's perspective. Thereby, relations between different use cases ensure a proper system understanding (Pohl & Rupp 2021).

The identification of use cases described in natural language requirements is a task of Requirements Engineering (RE) that is perceived as time-consuming for complex products like vehicles, and thus, offers potential for automation. The system architect can be supported with the help of Artificial Intelligence (AI) in combination with Natural Language Processing (NLP). Beyond that, the application of AI approaches is increasing in the field of RE. This includes AI-based pipelines for requirements elicitation, classification, and analysis (Dalpiaz & Niu 2020), supporting the product developer in general.

This publication contributes towards model-based requirements by providing a formalization and grouping pipeline that derives use cases based on the requirements specification on the system level. The remainder of this article is structured as follows: Section 2 illustrates the background and related work in this field. Section 3 presents the research need and resulting research question of the publication. Section 4 proposes a novel AI-based pipeline to solve this demand. Section 5 demonstrates the results of the pipeline with datasets from the industry. A critical discussion of the results is provided in Section 6. Lastly, Section 7 gives a summary and an outlook for future work.

2. Background and related work

The application of AI in RE is versatile and utilizes a broad variety of different NLP-based methods. Against this background, the following sections introduce relevant methods of NLP and related research work relevant for this article.

2.1. Methods

The presented pipeline builds upon fundamental concepts of NLP and AI. Thus, Section 2.1.1 introduces the concept of embeddings, which are used to represent natural language in a vector format. The fundamentals of clustering, which can be used to group requirements, are presented in Section 2.1.2. In addition, Section 2.1.3 introduces an NLP method to increase the accessibility of natural language documents.

2.1.1. Embeddings

Word or sentence embeddings are a common way to formalize natural language requirements into a machine-readable format by a vector representation. According to Jurafsky & Martin (2023), early embedding models are so-called static, as each word is represented by a fixed embedding which is learned once. This includes the Term Frequency-Inverse Document Frequency (TF-IDF) model that consists of two parts. The first part (TF) counts the frequency of a word within a document. The second part (IDF) assigns greater weight to words that appear in fewer documents. The combination results in a sparse vector, which represents the TF-IDF embedding.

A more sophisticated static embedding model is the skip-gram algorithm, which is part of word2vec (Mikolov *et al.* 2013). In contrast to TF-IDF, word2vec

https://doi.org/10.1017/dsj.2025.10019 Published online by Cambridge University Press

results in a dense vector. The embeddings are trained by considering the neighbouring words of the target word (Jurafsky & Martin 2023).

Contextual embedding models are characterized by having a distinct vector for each meaning of a word (Jurafsky & Martin 2023). Such word embeddings can be derived from the Bidirectional Encoder Representations from Transformers (BERT) model by Devlin *et al.* (2019). The model is based on a bidirectional transformer encoder architecture. To use a BERT model for a specific task, it can be fine-tuned based on pre-trained parameters (Devlin *et al.* 2019). The output of the BERT model is the word embedding and a special token that comprises the embedding for a given input sentence (Jurafsky & Martin 2023).

Sentence-BERT (SBERT) by Reimers & Gurevych (2019) builds upon BERT and specializes in sentence embeddings. Therefore, a Siamese network architecture is utilized. SBERT embeddings have a fixed length. SBERT can also be fine-tuned to a specific task-like information retrieval or document clustering. (Reimers & Gurevych 2019).

The latter task requires a similarity measure to evaluate two sentences based on their embeddings. Cosine similarity is a common metric for this purpose. Mathematically, it is equal to the normalized dot product of two vectors or embeddings, respectively (Jurafsky & Martin 2023).

In the context of RE, embeddings are used to represent linguistic and semantic information for further automated processing (Sonbol *et al.* 2022). In this context, highly similar embeddings could indicate redundant requirements.

2.1.2. Clustering

The use of clustering algorithms is a common approach to finding structures in vector data, such as embeddings. It is an unsupervised AI method in which a distance metric is used to determine whether two points belong together. The aim is to seek an assignment in which the distance within a cluster is minimized, whilst maximizing the distance between different clusters (Ertel 2017). There are algorithms that assign data points exclusively to one cluster, whereas overlapping clustering allows to assign a single data point to multiple clusters (Fortunato 2010). The cluster count is either known in advance or estimated by heuristic methods during the clustering process (Ertel 2017).

As many structures can be described as graphs, dedicated research exists to determine clusters within graph data. A graph consists of nodes and edges connecting the nodes. If all edges are unordered pairs, the graph is undirected. In a weighted graph, each edge is assigned a weight (Schaeffer 2007). Similar to clustering of vector data, graph-based clustering aims to find structures in which nodes have a high probability of belonging together. It is especially common in the field of bioinformatics and network analysis (Fortunato 2010).

Altaf-Ul-Amin *et al.* (2006) propose a graph-based clustering algorithm to identify protein interactions. It is applied to an undirected graph and utilizes weights. The edge weights represent the number of shared neighbours of connected nodes and are merged individually for each node. This approach can be expanded to generate overlapping clusters. Further, the clustering algorithm can be adapted to graphs from other domains apart from protein interaction.

In contrast, Shchur & Günnemann (2019) develop their algorithm for domain-independent applications from the outset. The focus is on overlapping clustering of

https://doi.org/10.1017/dsj.2025.10019 Published online by Cambridge University Press

undirected unweighted graphs. The algorithm is called Neural Overlapping Community Detection (NOCD). For this, a Graph Neural Network (GNN) is utilized. The GNN uses a matrix representation of the non-clustered graph as input and outputs a threshold-based clustering of the nodes.

Often, there exists some labelled data that can be integrated in the clustering process, which leads to Semi-Supervised Clustering (SSC). It complements the unlabelled data of traditional clustering. A common way to represent the labelled data is pairwise must-link and cannot-link constraints (Cai *et al.* 2023). Zhao *et al.* (2012) propose an SSC algorithm for document clustering. The algorithm can deal effectively with the interference of clusters but is not overlapping according to the before given definition.

Different metrics are available to evaluate the results from clustering. According to Amigó *et al.* (2009), intrinsic metrics are best suited for an unlabelled dataset. Here, intra- and inter-cluster similarity are investigated without reference to ground truth data. For a dataset with ground truth, extrinsic metrics can be utilized. This usually requires manual effort for labelling the dataset. Extended BCubed (Amigó *et al.* 2009) is an evaluation metric considering overlapping clustering. The metric comprises a precision and a recall value. Precision decreases if more clusters than necessary are identified. This equals more information than in the ground truth data. The recall decreases if items with multiple labels are not represented in multiple clusters. This means that too little information was created. The F-score is the harmonic mean of precision and recall.

The application of clustering plays an important role in AI-based RE because it can be used to structure large requirements specifications (Sonbol *et al.* 2022).

2.1.3. Keyphrases

Keyphrases can be used to make clusters of natural language data more accessible, as they provide a context. In general, there are supervised and unsupervised keyphrase extraction methods (Ajallouda *et al.* 2022). The advantage of an unsupervised method is that no labelled training data is necessary. However, the accuracy is usually lower than that of supervised approaches (Schopf *et al.* 2022). The process of keyphrase extraction comprises the identification of candidate keyphrases, usually by utilizing Part-Of-Speech (POS)-tagging (Ajallouda *et al.* 2022). Grootendorst (2020) introduces a word n-gram-based framework to extract keyphrases. Here, the ranking of candidates is based on an SBERT model. Schopf *et al.* (2022) expand this approach by using POS-tagging for the extraction part. This results in the unsupervised PatternRank model

The primary benefit of RE is the improved accessibility of requirement sets with unknown content. This is achieved by describing them based on their core aspects

2.2. Related work

In RE, NLP is applied to a large variety of different tasks. Among others, Sonbol *et al.* (2022) provide a comprehensive literature review concerning different scenarios of NLP-based methods in RE. The authors identify five general tasks comprising management, modelling, extraction, quality, and analysis. The last task comprises most of the contributions and is further subdivided into traceability, classification, semantic role labelling, and clustering.

https://doi.org/10.1017/dsj.2025.10019 Published online by Cambridge University Press

A first approach in the field of clustering is proposed by Casamayor *et al.* (2012). The authors propose a semi-automatic framework for grouping functionally related requirements within the scope of software engineering. For this, methods of text mining are used to identify architectural elements that result in the responsibilities of the system. As similar responsibilities should be implemented in one conceptual component, an unsupervised clustering algorithm is applied. Hence, this publication contributes to a functional decomposition of a requirements specification.

Similarly, Misra *et al.* (2016) extract topics of a requirements specification to enhance the system understanding by decomposition. However, the natural language requirements are pre-processed and represented by a simple vector space model. Latent semantic analysis measures the similarity between requirements and extracts themes from the requirements specification. A clustering step assigns every natural language requirement to one or more identified themes.

Mokammel *et al.* (2018) describe a framework beginning with the automated extraction and refinement of requirements. This is followed by a clustering step. For this, a keyword-requirement matrix is established and processed with latent semantic analysis. Similar requirements are then clustered with K-Means. Additionally, the clusters are named based on the previously created matrix. Parallel to this, a second matrix containing the cosine similarity of requirement pairs is created and used to analyse the similarity further.

Salman *et al.* (2018) use clustering to decompose large requirement documents to distribute smaller groups of requirements to different development teams. For this, each requirement is pre-processed by removing frequent words and converted into a vector space model. A simple count-based vector representation is used for this purpose. Agglomerative hierarchical clustering combined with cosine similarity is applied to retrieve the clusters. The optimum number of clusters is automatically determined.

In addition, Gülle *et al.* (2020) aim to increase the accessibility of crowdsourced user stories. For this, topic modelling is utilized and allows for deriving requirements from user stories more easily. The authors report best results with a combination of word2vec embeddings, as a more sophisticated embedding model compared to the previous approaches, and word mover's distance (Kusner *et al.* 2015) as a similarity measure.

Furthermore, Kochbati *et al.* (2021) cluster related user stories, which are high-level descriptions of requirements. Similar to Gülle *et al.* (2020), the word2vec model calculates word embeddings, which are combined to requirement embeddings by a scoring function that also computes the pairwise similarity of requirements. Based on this, hierarchical agglomerative clustering is utilized with an automated determination of the optimum number of clusters. Additionally, heuristics are established to derive use case diagrams from the clustered user stories, whereas each user story leads to a single use case within the diagram.

Lastly, Bisang *et al.* (2022) aim to identify redundant natural language requirements and thereby contribute to the research topic of finding similar requirements. The authors focus on requirements that are formulated in different languages. For this, multilingual language models are compared with the combination of machine translation and a monolingual embedding model. The results indicate a better performance of multilingual language models. This is potentially explained by very specific words and formulations in the requirement domain.

https://doi.org/10.1017/dsj.2025.10019 Published online by Cambridge University Press

3. Research question and methodological approach

The importance of MBSE is growing steadily in the automotive industry and other sectors. A possible approach starts with Model-based Requirements Engineering (MBRE). Here, an early step is modelling the system context based on provided requirements specifications, for example, from previous stakeholder analysis, in the form of use case diagrams. Each use case documents a system functionality and identifies relationships to both actors, external systems, and other use cases. By this, the overall system understanding is increased. For this, it is necessary to decompose the requirements specification (Pohl & Rupp 2021). Thus, the identification of use cases from interrelated natural language requirements is an important task in industrial application (Schleifer *et al.* 2024).

As shown in the previous section, there are different approaches within the context of this overall challenge. The initial deployment of semi-automatic approaches such as Casamayor *et al.* (2012) has been followed by an increasing utilization of AI methods. For example, Kochbati *et al.* (2021) propose a pipeline from natural language requirements to use case diagrams using word embeddings and clustering. The conceptual framework of Schleifer *et al.* (2024) aims to cover the context modelling by use case diagrams specific for the automotive industry. This framework consists of two steps. The first step is the analysis and structuring of the given requirements specification, that is, the use case identification. This highlights the importance of decomposing requirements specifications into use cases. This step is followed by the derivation of the use case diagram with its relationships, see Figure 2.

Nevertheless, neither of the previously introduced approaches fully covers the complexity of the industrial demand, which arises when identifying use cases from natural language requirements. In globally operating companies, it is common that the requirements are not monolingual, as it is typically required by language models of NLP. To this point, the automatic identification of use cases from multilingual requirements specifications has not been considered. Requirements on the system level often have varying levels of detail and are potentially non-atomic. This means that a single requirement can address multiple aspects of functionality. This requires a non-exclusive mapping. By that, one requirement can be grouped into multiple use cases and contributes to each of them. Following the example proposed in Schleifer *et al.* (2024), the requirement 'The vehicle shall provide a remote unlock functionality' contributes to two use cases, namely 'unlock vehicle' and 'keyless access'. Existing approaches neglect this necessity. Further, a suitable approach for industrial applicability must be able to work with engineering knowledge from ongoing development. In this publication, available development

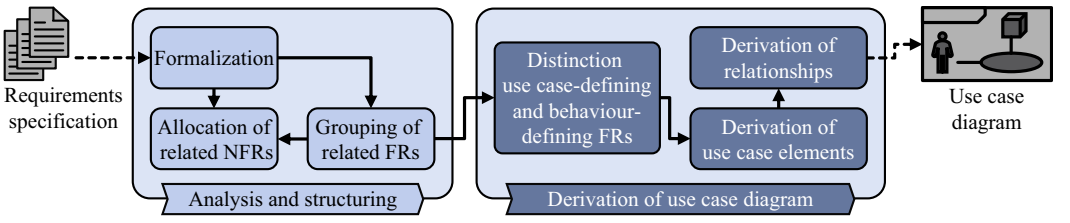

Figure 2. Framework for use case diagram derivation according to Schleifer *et al.* (2024) including Functional Requirements (FRs) and Non-Functional Requirements (NFRs).

https://doi.org/10.1017/dsj.2025.10019 Published online by Cambridge University Press

knowledge is understood as manually created mappings between requirements and use cases. In the early stages of development, this information is sometimes already available due to transfers from earlier development or preliminary manual work. However, these mappings are most likely incomplete. Additionally, an approach with industrial applicability must work both in an unsupervised and semi-supervised manner. Existing approaches are either supervised or unsupervised, though. Consequently, there are currently two major challenges that have to be overcome:

- Non-exclusive grouping of requirements in use cases, and
- Incorporation of available development knowledge, if existing.

This leads to the following research question:

How can AI-based methods of NLP be used to overcome the current challenges in identifying use cases from a natural language requirements specification?

To address this question, this publication proposes a novel modular AI-based pipeline with two main steps. First, methods from NLP are used to establish a requirements graph. If prior knowledge exists, it is integrated into the graph and serves as semi-supervision. Second, an overlapping graph clustering algorithm is applied to derive potential use cases. The accessibility is increased by providing keyphrases to each use case. The pipeline is implemented as a prototype and tested with different sets of input requirements from the given automotive use case to validate its applicability. The results are evaluated on the basis of different levels of existing supervision of ongoing development. Further, manual analysis by experts from the industry is conducted.

Throughout the following sections, an automotive case study illustrates the individual steps of the proposed pipeline. The subject of this study is a requirements specification belonging to a sample feature called 'passenger comfort'. This specification comprises 61 requirements. Excerpts of the resulting use cases can be found in Table 1 on page 16, as this feature also serves for the concluding experiment.

4. Pipeline

With reference to the research gap stated in Section 3, a comprehensive pipeline for identifying potential use cases described in a system requirements specification is proposed. Figure 3 illustrates the successive steps, which are briefly presented in this paragraph and further elaborated in the following subsections.

At the beginning (refer to Section 4.1), as part of the **pre-processing**, it is necessary to remove syntax errors from the natural language requirements specification to get a more homogeneous input document. This step includes the translation in case of multilingual requirements to receive a monolingual set of requirements. Afterwards, Section 4.2 describes the representation of each requirement by a node in the **requirements graph**. The information enclosed in the natural language requirements is extracted and formalized with the help of an NLP pipeline. This includes the fine-tuning of pre-trained language models, the generation of sentence embeddings for each requirement, and the determination of similarities based on the embeddings. This results in edges linking the requirements. If available, the graph is supplemented by edges based on existing development knowledge. This comprises an early, incomplete state of manual mapping of functionally related requirements from current development projects. In order

https://doi.org/10.1017/dsj.2025.10019 Published online by Cambridge University Press

Table 1. Extract of the results of a feature without extensive ground truth data

Use case 1	
Keyphrase	Operate the passenger seat
Req. 1	The vehicle system must offer the user the possibility to adjust the longitudinal position of the front passenger seat.
Req. 2	The vehicle system shall offer the user the possibility to adjust the seat height of the front passenger seat.
Req. 3	The vehicle system shall offer the user the possibility to adjust the side bolster of the driver's seat.
	…
Use case 2	
Keyphrase	Adjust the seat height
Req. 4	The vehicle system shall offer the user the possibility to adjust the seat height of the driver's seat.
Req. 3	The vehicle system shall offer the user the possibility to adjust the side bolster of the driver's seat.
	…
Use case 3	
Keyphrase	Store the last steering column position
Req. 5	The vehicle system shall offer the driver the possibility to adjust the steering column electrically.
Req. 6	The vehicle system must store the last steering column position used by the driver.
Req. 7	The vehicle system shall offer the driver the possibility to load the last stored steering column position.

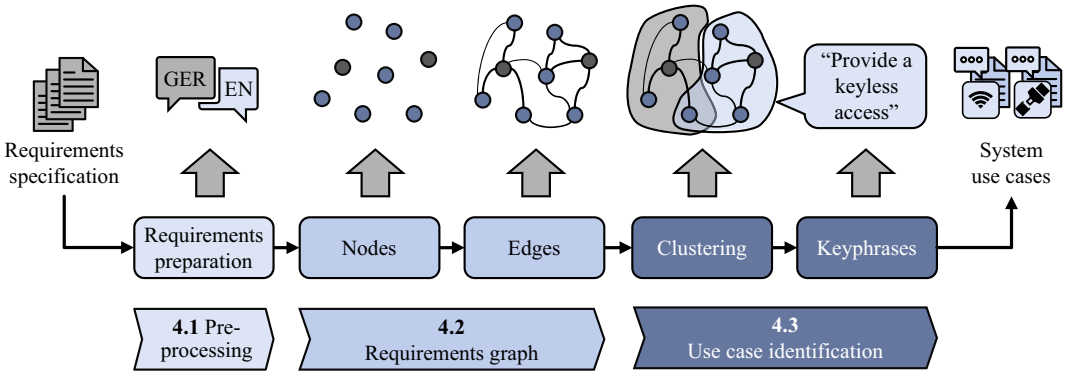

Figure 3. Overview of the proposed pipeline for grouping natural language requirements to system use cases.

https://doi.org/10.1017/dsj.2025.10019 Published online by Cambridge University Press

to unify the requirement links, which are automatically derived from the already existing, explicitly defined ones, a semi-supervised approach is proposed. Subsequently, the generated graph allows the **use case identification** based on requirements, which refer to similar functionalities by utilizing an overlapping graph clustering algorithm (refer to Section 4.3). The results are enriched by a keyphrase extraction for each cluster, which supports the engineer with the evaluation and the final use case definition.

4.1. Pre-processing

The necessary pre-processing steps are elaborated in Figure 4. A requirements specification typically comprises additional attributes and entry types like headlines and information items besides the natural language requirement text (Pohl & Rupp 2021). The input for the pipeline is a filtered requirements specification that only comprises Functional Requirements (FRs) and Non-Functional Requirements (NFRs). These requirements might not strictly adhere to a template. Besides that, it appears that some requirements contain syntax errors like missing full stops or misplaced line breaks. An automatic rule-based **removal of syntax errors** based on a predefined sentence pattern is part of the proposed framework. The next step is transforming a potentially multilingual requirements specification into a monolingual one. For this, an **automated translation** based on AI is utilized. An available software tool with a domain-specific corpus for the automotive industry executes the translation. These steps perform better in conjunction with the prior text clean-up. In addition, a manual **random sample check** is recommended to guarantee the quality of the translated requirements. This is the only manual intervention during the pre-processing and allows early detection of deficiencies if the automated translation.

For example, a syntactically incorrect requirement formulated in German might be: 'Das Fahrzeugsystem muss die letztgenutzte\ nLenksäulenposition des Fahrers speichern!'. Here, it can be seen that the requirement ends with an exclamation mark instead of a full stop. '\ n' indicates an unwanted line break. The pipeline resolves both issues and translates the requirement into English. The resulting requirement is as follows: 'The vehicle system shall store the last steering column position used by the driver.'

4.2. Requirements graph

The generation of the requirements graph forms the essential part of natural language formalization. Due to the semi-supervised approach of this pipeline,

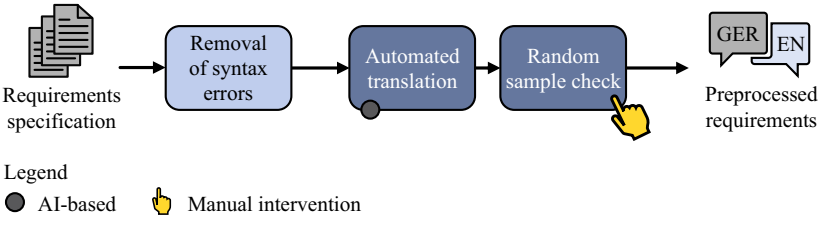

Figure 4. Detailed view of the pre-processing steps.

https://doi.org/10.1017/dsj.2025.10019 Published online by Cambridge University Press

the requirements graph allows to incorporate available development knowledge in the form of existing links between functionally related requirements. This information is provided to the pipeline as a table. As depicted in Figure 5, the input for this stage is the preprocessed requirements. The generation of the graph is divided into node (light blue) and edge (dark blue) definitions. The requirements graph represents the output of this step and its structure allows to use of different existing clustering algorithms. This allows to choose existing clustering algorithms tailored to specific problems. FRs and NFRs are both used as input data for the graph generation.

4.2.1. Definition of nodes

The left side of Figure 5 illustrates the pipeline steps concerning the definition of the nodes (light blue) in the requirements graph. First, a **pre-trained language model** is loaded. This is the basis for the requirement embeddings. Generally, publicly available language models are domain-independent as their training corpus is versatile and accordingly not aligned with automotive system requirement formulations. In order to obtain a language model that is better tailored to the task at hand, **fine-tuning of the language model** may be carried out. This is only possible if the ongoing development offers existing knowledge in the form of data, which can be used to guide the original language model into a more domain-specific one. The required training data is manually created by the system architect. Here, links based on functional relations between requirements are drawn and typically documented in a table view. Such links are likely to exist as part of the initial manual screening of the requirements specification, or from previous development projects. To derive the training examples for the fine-tuning step, two related requirements are paired together. By iterating over the already linked requirements, the complete training corpus is constructed. This approach reveals positive training examples only, which correspond to must-link constraints. The nature of the existing development data (training data) makes it impossible to

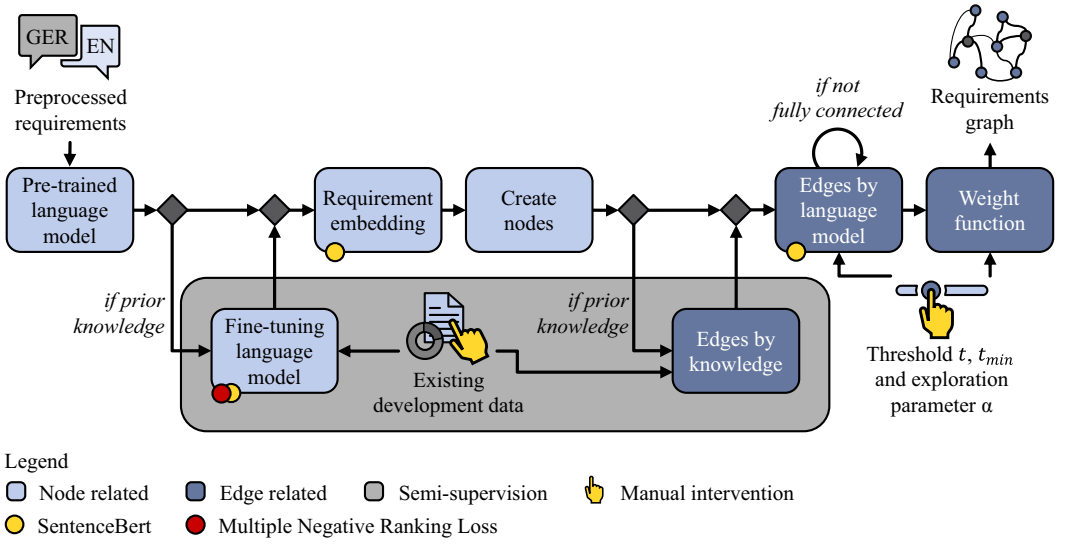

Figure 5. Formalization of a requirements specification with a requirements graph.

https://doi.org/10.1017/dsj.2025.10019 Published online by Cambridge University Press

derive cannot-link constraints, as a non-existent link between two requirements does not automatically imply a contradiction. Instead, it is assumed that the current state of development is not final and additional must-link constraints or functional similarities can exist, respectively. For this reason, the language model is fine-tuned with SBERT and the loss function Multiple Negative Ranking Loss (MNRL), as this loss function is suited for training tasks with only positive examples in particular. By this, SBERT learns to increase the similarity of embeddings corresponding to the pairwise linked requirements from the training corpus. This results in a language model, which is more suited to represent the domain-specific characteristics of the requirements specification.

The next step calculates the **requirement embeddings**. Depending on the availability of existing development knowledge, either the initial language model or the fine-tuned one is utilized to transform the natural language requirement texts into requirement embeddings. SBERT calculates the embeddings for each requirement automatically. This results in a dense vector representation, which is always of the same dimension, independently of the input text length. The last step is to **create the nodes**. Each requirement is represented by a single node in the requirements graph. The natural language requirement text, embedding, and the type of the requirement (FR or NFR) are stored as parameters of the node.

Following the case study, a pre-trained language model is fine-tuned based on existing links between requirements. For example, requirements 1 and 2 from Table 1 are already mapped to a common functionality by the system architect. Afterwards, each requirement text is transformed into an embedding. The requirements graph of the case study comprises 61 nodes after this step.

4.2.2. Definition of edges

The edges of the requirements graph represent the functional similarity between pairs of input requirements. The extent of this similarity can be described further by edge weights. All edge-related steps are marked in dark blue in Figure 5.

If prior knowledge from existing development data exists, a rule-based approach is utilized to represent this explicit link between two requirements in the requirements graph. The **edges of knowledge** are represented by the edge weight $w_p = 1$. By this, it is guaranteed that a must-link relation is present in the requirements graph. In addition, the existing development data influences the edges drawn by the language model, as they were used for fine-tuning. This leads to the second source of edges.

SBERT is utilized to derive **edges by the language model** from the previous step. A common way to estimate the similarity of two embeddings is the cosine similarity (Reimers & Gurevych 2019). It calculates the cosine of the angle between two embedding vectors in their hyperspace. If this results in a value exceeding the user-defined threshold t for a pair of requirements, then an edge is established between the two corresponding nodes. The graph is not necessarily fully connected after iterating over all requirement pairs. However, a fully connected graph is essential for the subsequent use case identification. For this reason, the threshold for creating new edges between the nodes of the requirements graph is incrementally decreased until there are no more isolates. A break condition in the form of a manually chosen minimal threshold t_{min} limits the overall number of edges. If the

https://doi.org/10.1017/dsj.2025.10019 Published online by Cambridge University Press

graph is not fully connected before reaching t_{min}, remaining isolated nodes are neglected as they are considered to be too dissimilar to the rest of the requirements specification. Such requirements need to be complemented manually to the resulting use cases at the end. The edge weight based on the requirement embeddings and the SBERT model is indicated by w_e. As the weight w_e of the initial edge creation is retained, the iterative approach does not reduce the weight of existing edges.

A combination of the two edge weights w_p and w_e is necessary as most clustering algorithms consider only one edge weight. For this reason, a **weight function** is introduced:

$$w = \alpha \cdot w_p + (1 - \alpha) \cdot w_e \tag{1}$$

By this, a combined edge weight w is calculated. Additionally, the resulting weights are normalized by division with the largest weight value w_{max}. Equation (1) introduces the exploration parameter α, which allows the user to influence the subsequent clustering. By choosing values larger than 0.5, the combined edge weight emphasizes the rule-based edges from the existing development data. Conversely, a focus on the language model results from values for α smaller than 0.5. This means that even if there is a lot of supervision through existing development data, its explicit influence can be reduced ($\alpha < 0.5$) and a new, explorative mapping based on the language model can be generated. Thus, the existing knowledge only has an indirect influence on the result through fine-tuning. For example, the nodes of the case study are connected with the exploration parameter equal to 0.5 to incorporate existing development knowledge and similarities between embeddings equally.

4.3. Use case identification

Figure 6 illustrates the individual steps of the use case identification. The requirements graph serves as input for the clustering step, which corresponds to the framework step 'grouping related FR' depicted in Figure 2. Extending Schleifer *et al.* (2024), both FR and NFR are considered here. The clustering itself is not semi-supervised but takes into account the industrial demand for overlapping clusters. Thereby, a large number of graph clustering algorithms can be utilized, whereat most originate from the bioinformatics domain or network analysis. With the help of keyphrase extraction and ranking, it is possible to increase the accessibility of the use cases for the system architect. The accessible use case mappings represent the overall output of the pipeline.

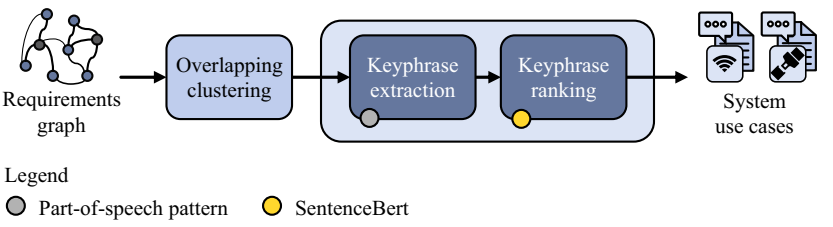

Figure 6. Detailed view of the use case identification.

https://doi.org/10.1017/dsj.2025.10019 Published online by Cambridge University Press

4.3.1. Clustering

The pipeline employs an **overlapping clustering** algorithm to identify use cases based on the requirements graph. Graph-based clustering algorithms, established in research fields such as bioinformatics and network analysis, are deemed to be suitable for this purpose. Since the number of use cases described in a requirement set is unknown in advance, the clustering algorithm must be able to determine the optimal cluster count at runtime. This is especially important if existing development data contributed to the graph generation. In this case, the system architect manually identified some potential use cases, but there might be additional ones in the requirement set. Likewise, it can be beneficial to combine some manually identified use cases. This leads to less clusters than provided for the graph generation as described in Section 4.2. Further, the clustering algorithm must be based on edge weights rather than node connectivity, as the edge weights resulting from Equation 1 are an integral part of the pipeline. Otherwise, the manual influence of the exploration parameter α is lost. This parameter allows the system architect to influence the results of the clustering algorithm without additional training.

The identified clusters, that is, use cases, of the case study are shown in excerpts in Table 1 and discussed in detail in Section 6. It can be seen that functionally similar requirements belong to one cluster. As expected, requirements 1 and 2, previously assigned as examples of existing development knowledge, share a cluster.

4.3.2. Keyphrases

The results of the graph clustering form the basis for identified use cases and are presented to the system architect. To increase the accessibility and facilitate the manual examination of the proposed use cases, each cluster is described by multiple keyphrases. For this, PatternRank (Schopf *et al.* 2022) is used for **keyphrase extraction** from the requirements within a cluster. A custom POS pattern is defined

$$((<VB>|<VBZ>|<VBP>|<VBD>|<VBG>|<VBN>) \\ +(<.*><JJ>*<NN>+))$$

The first part matches a single verb in various forms. This is combined with an optional word of any type, an optional adjective, and one or more nouns. If this POS pattern is matched in the requirements within the use case, the corresponding part of the requirement text is extracted as a candidate keyphrase. This can lead to multiple potential keyphrases for each use case. Thus, the descriptive phrases are ranked in decreasing order according to their similarity to the natural language requirements of the use case. The language model from Section 4.2 is used for the **keyphrase ranking**. The best matches are presented alongside the use case requirements to assist the system architect

The first identified use case of the case study deals with adjusting the passenger seat of the vehicle. The keyphrase 'Operate the passenger seat' improves the understandability

5. Application

The pipeline is implemented and tested with industrial datasets to evaluate its performance. Details of the specific implementation, the datasets, and the

https://doi.org/10.1017/dsj.2025.10019 Published online by Cambridge University Press

conducted experiments are given in Section 5.1. The results of the experiments are presented in Section 5.2 and discussed in Section 6.

5.1. Implementation and datasets

The pipeline is implemented in Python and tested on a laptop with an 11th-generation Intel i7 processor, 32 GB RAM, and a NVIDIA RTX A3000. A cloud computing environment is used in the case of labour-intensive fine-tuning of the language model. The pipeline uses the pre-trained language model 'all-MiniLM-L6-v2' (huggingface 2024) in combination with SBERT (Reimers & Gurevych 2019). In case of existing development data, the language model is fine-tuned with the derived training examples over 20 epochs. The clustering algorithm of Altaf-Ul-Amin *et al.* (2006) is adopted to use the combined edge weight from Equation (1) instead of deriving edge weights from the neighbouring nodes as originally. This adoption allows for manual influence on the clustering results by the exploration parameter α. The implementation by Rossetti *et al.* (2019) is utilized. The extraction of keyphrases follows the implementation by Schopf *et al.* (2022).

Two industrial datasets from the automotive domain are used to evaluate the pipeline. Each dataset is pre-sorted according to superordinate system functionalities. By that, a dataset contains multiple use cases, which need to be identified, from a certain scope of the vehicle system. These are 'human-machine-interaction' (dataset 1) and 'passenger comfort' (dataset 2) with 125 and 61 requirements, respectively. The first dataset is completely mapped by a system architect into 20 clusters. 46 out of 125 requirements belong to more than one cluster in this dataset. This solution corresponds to a possible ground truth data for evaluation. The second dataset has 39 initially mapped requirements. This equates to roughly 64% manually mapped requirements. To place this semi-supervised scenario closer to the intended usage, half of these mappings are neglected, which ultimately results in a semi-supervision ratio of approximately 32%.

The semi-supervision ratio r is introduced to describe the experiments carried out. Here, r denotes the percentage of mapped requirements in the requirements specification. This ratio is extended to the simulated semi-supervision ratio r_{sim} to describe the application of completely mapped datasets. Then, r_{sim} is the ratio of used training pairs that are incorporated during the generation of the requirements graph. This allows to use of metrics like extended BCubed for evaluation whilst only considering a fraction of supervision in the pipeline.

Three individual experiments are carried out based on this concept, see Figure 7. Experiments (a) and (b) examine the influence of the exploration

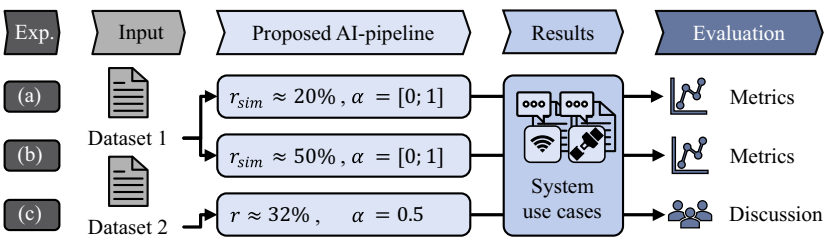

Figure 7. Overview of the conducted experiments (Exp).

https://doi.org/10.1017/dsj.2025.10019 Published online by Cambridge University Press

parameter α. Dataset 1 serves as input, and the extent of simulated semi-supervision is set to $r_{sim} = 20\%$ and $r_{sim} = 50\%$ for experiments (a) and (b), respectively. The resulting system use cases are evaluated by extrinsic metrics. Experiment (c) investigates the pipeline with dataset 2, and thus, without ground truth data. The exploration parameter is set to $\alpha = 0.5$. The results are evaluated manually by discussion with experts.

5.2. Experimental results

The resulting metrics for experiments (a) and (b) are shown in Figure 8. The value range of the exploration parameter α is plotted on the x-axis in both cases. Four evaluation metrics are derived from the results, beginning with the F-Score as the harmonic mean of extended BCubed precision and recall. Next, the ratio of resulting overlapping requirements to ground truth overlaps is considered. In addition, the percentage of requirements that are lost during the clustering step is calculated. Lastly, the ratio between identified clusters by the pipeline and to ground truth number of clusters is used to evaluate the experiments.

Experiment (a) with 20% simulated semi-supervision shows an F-Score between 0.7 and 0.6 up to $\alpha \approx 0.5$ with a significant drop to 0.4 for larger values of α. The ratio between actual and ground truth overlapping requirements is between 0.4 and 0.2 decreasing with larger α values. The ratio of requirements that are lost by the clustering algorithm increases from approximately 10% for $\alpha =$ 0 to 20% $\alpha = 1$. Lastly, the ratio of identified clusters is around 0.8 for $\alpha = [0;0.4]$, which means less clusters than in the ground truth data. The cluster count enlarges with increasing α up to 1.4 of the ground truth cluster count.

Experiment (b) incorporates 50% simulated semi-supervision. The results show a stable F-Score metric around 0.8 for the entire value range of α. The ratio of overlapping requirements is between 0.2 and 0.4 which corresponds with less overlap than in the ground truth data. The ratio of lost requirements during the clustering step is generally low but increases up to approximately 0.1 for larger values of α. The ratio of the number of identified clusters compared to the ground truth number of clusters is smaller than 1 for all values of α, meaning that less use cases are derived. However, an increase with increasing α is noticeable.

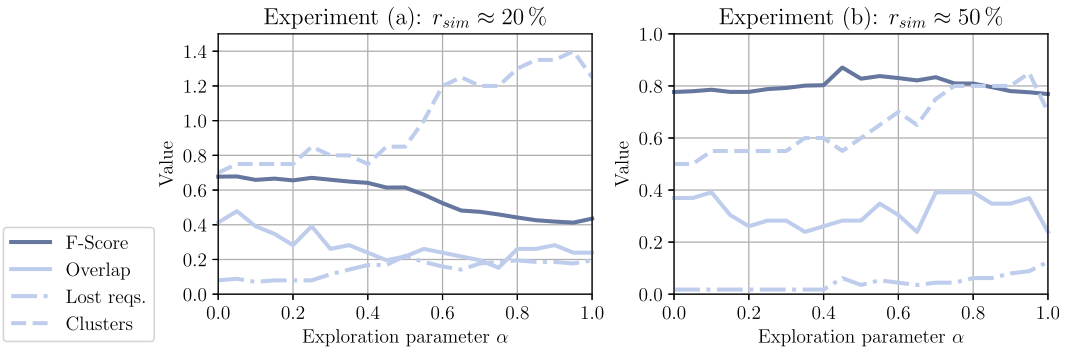

Figure 8. Metrics for a feature with extensive ground truth data for two different scenarios: $r_{sim} \approx 20\%$ and $r_{sim} \approx 50\%$ simulated semi-supervision ratio.

https://doi.org/10.1017/dsj.2025.10019 Published online by Cambridge University Press

In experiment (c), the pipeline was tested on the second set of requirements (dataset 2) without comprehensive ground truth data in the form of a finished manual use case identification. Thus, it was applied in the intended manner. The evaluation for the last experiment is based on an open discussion with experts from the industry. The input requirements specification used was thoroughly examined before assessing the resulting use case clustering. Any drawbacks of the pipeline were duly noted and are discussed in the next section. Excerpts of the results of this experiment can be seen in Table 1. Here, the industrial requirements specification is clustered by the proposed pipeline according to use cases. Three of five use cases are shown with exemplary requirements for each cluster.

6. Case study and discussion

This section provides a thorough examination of the implications of the findings from the proposed pipeline. First, the task of identifying use cases described in interrelated natural language requirements is a complex task, and thus, can not be measured solely by a single extrinsic metric. Unlike classification or clustering problems with unambiguous ground truth, assigning requirements to use cases is more complex. There are various possible functional groupings. For instance, in the automotive context, requirements can be grouped based on vehicle functionalities or operating modalities. The fact that different requirement groupings are possible means that a lower result in the F-Score against a potential ground truth standard does not necessarily equate to poor performance. For this reason, an additional manual assessment of the resulting clustering is carried out in a second step. Nevertheless, the results from Figure 8 help to assess the functionality of the pipeline in general.

For experiment (a) with $r_{sim} \approx 20\%$, that is, consideration of little existing development data, a shift towards larger values for α, and thus, focusing existing mappings, leads to a decreasing F-Score. Conversely, this shows the positive influence of the requirements embeddings created by the SBERT model. If the amount of prior knowledge increases ($r_{sim} \approx 50\%$), this decrease is absorbed and results in a stable F-Score measure. The demand for overlapping clusters is an essential contribution of this pipeline. By reviewing the ratio between ground truth overlaps and clustered overlaps, it becomes apparent, that the pipeline results in too little overlapping requirements in both cases $r_{sim} \approx 20\%$ and $r_{sim} \approx 50\%$ as the corresponding metric is smaller than 1, see Figure 8. In the second scenario, this gap is decreasing for larger values of α, which shows that adding labelled data increases the performance if the focus of the pipeline is set on manually drawn links between requirements. This gap can potentially be reduced by adapting the hyperparameters of the exerted clustering algorithm by Altaf-Ul-Amin *et al.* (2006) or utilizing different algorithms. A crucial observation is that up to 20% of the requirements are lost during clustering. This is most likely caused by the functionality of the clustering algorithm. For this reason, different clustering techniques could further increase the performance of the pipeline. Up to now, lost requirements during the use case identification step result in additional work. The system architect has to manually assign the remaining requirements to the proposed clusters afterwards. Lastly, the number of identified clusters is either too small or exceeds the ground truth data. Especially for scenario $r_{sim} = 20\%$ a significant step around $\alpha = 50\%$ can be observed. Since the extended BCubed

https://doi.org/10.1017/dsj.2025.10019 Published online by Cambridge University Press

metric favours cluster quality over quantity, the parallel decrease of the F-Score can be explained. The trend towards convergence for increasing α in scenario $r_{sim} = 50\%$ shows that a focus on prior knowledge with simultaneously more supervision leads to better results of the pipeline proving the applicability. In general, the results can be adjusted towards the specific needs by testing different hyperparameters and clustering algorithms.

As mentioned earlier, a sole focus on ground truth data from one possible clustering result is not sufficient. For this reason, Table 1 shows extracts of the identified use cases for a different set of requirements concerning passenger comfort functions. The results were discussed with experts from the industry. The pipeline results in a use case clustering by passenger seat (use case 1), driver seat (use case 2), and steering column (use case 3). The first two use cases show excerpts of the associated requirements, whereas the third use case is shown completely. The keyphrase indicates the scope of the passenger seat correctly for the first use case. For use case 3, the keyphrase hints correctly at the steering column. In case of the second use case, the keyp hrase is not suitable to assist the system architect with an appropriate headline. Here, the keyphrase suggests a resulting mapping by seat movements instead of different seating areas within the vehicle. The assignment of requirement 3 in use case 1 is incorrect based on the given scope. However, the requirement is clustered to use case 2, which concerns the driver's seat, as well. When analysing the requirements as a whole, it becomes clear that no overlapping allocation is necessary for this specific requirement set. With a few exceptions, this is recognized correctly by the algorithm. Use case 3 is an example of a complete and correctly assigned cluster. It contains all requirements concerning the steering column. Requirements 1 and 6 illustrate a problem when utilizing an AI-based translation tool. In the automatic translation, the German signal word 'müssen' was correctly translated as 'must', but a domain-specific characteristic was not taken into account. It is common to use the verb 'shall' instead (Dick *et al.* 2017).

Summing up, the defined research question in the present contribution can be answered as follows: The combination of a semi-supervised requirements graph generation based on sentence embeddings with an overlapping clustering algorithm is a sound basis for meeting practical demands arising when identifying use cases from natural language requirements specifications. This can be seen at the external metrics presented in Figure 8. The F-Score is arguably not especially high, but, due to the absence of a single ground truth solution, a good indicator for a working pipeline. This initial assessment is confirmed by the manual evaluation of Table 1. With that, it can be stated that the utilization of AI-based translations is capable of preserving the functional substance of requirements. Further, the division into a semi-supervised requirements graph generation and the overlapping clustering allows to utilize existing manual allocations as well as considering different levels of detail concerning the requirements. In contrast to the existing research presented in Section 2.2, this pipeline is fully automated and only requires the user to perform an optional check of the AI-based translation and set the parameters α, t and t_{min}. However, a manual check of the overall results is recommended. By considering the practical demands for overlap and semi-supervision, the pipeline is suited to support a system architect to identify use cases from requirements specifications more efficiently. The combination of a pre-trained language model with the development knowledge of multiple system

https://doi.org/10.1017/dsj.2025.10019 Published online by Cambridge University Press

architects enables the identification of previously unidentified use cases. In addition, the bias caused by the manual work of individual developers is reduced, as the knowledge of multiple system architects can be included in the requirements graph. Furthermore, the pipeline shows good performance even with low computational resources like an ordinary laptop computer. Each presented experiment was calculated in <10 min.

Nevertheless, limitations arise due to the industrial dataset. This limits the replicability of the results for related research. Further, the experimental results from Section 5 show general limitations of the specific implementation of the proposed pipeline. The problems relate to the loss of requirements, too few overlapping requirements and the number of clusters. This can potentially be avoided by utilizing different clustering algorithms. Initial tests using the clustering algorithm by Shchur & Günnemann (2019) show promising results. The loss of requirements is avoided, and thereby the manual effort is further reduced. In addition, the number of overlapping clusters comes closer to the ground truth data. To increase the overall clustering performance, modifications to the requirements graph are suggested. At this point, the advantages of the graph-based approach come into effect. Additional sources of knowledge between pairs of requirements can be easily integrated into the graph as long as a weight-based representation is possible. An example therefore is a rule-based assessment of the natural language requirements based on vocabulary or sentence structure. By adopting the weight function to new information sources, the influence on the clustering algorithm results is preserved. Further, the division in semi-supervised knowledge representation and unsupervised overlapping clustering allows to utilize a broad variety of clustering algorithms and enables profiting from future developments in this research area. Keyphrases based on POS patterns show promising results in increasing accessibility. The generation of keyphrases can potentially be enhanced by utilizing a generative large language model as shown in Song *et al.* (2024). Lastly, limitations to the evaluation arise due to the unstructured assessment by experts from industry. This is sufficient to gain a general understanding of the pipeline's capabilities, but lacks in serving as a final evaluation.

7. Conclusion and outlook

The identification of use cases in large sets of natural language requirements is an important step towards model-based requirements. Until now, meeting the specific needs outlined above has required a time-consuming manual process. Against this background, the publication contributes towards the automatic identification of use cases based on natural language requirements. For this, AI methods are combined to a novel pipeline. The pipeline is divided into two parts. First, a requirements graph is established, incorporating prior knowledge from development. This step is realized with an SBERT model. Second, overlapping clustering algorithms identify individual use cases and their corresponding requirements. The proposed pipeline is complemented by experimental results for two different sets of requirements. The results confirm the applicability of the pipeline by meeting the practical demand for handling multilingual requirements, consideration of existing manual assignment, and overlapping clustering of requirements in use cases. Consequently, this allows for a more comprehensive and less subjective system understanding by the system architect. The majority of the requirements

https://doi.org/10.1017/dsj.2025.10019 Published online by Cambridge University Press

specification is automatically converted in use case proposals with only a little manual work remaining. In general, the pipeline forms the basis for the subsequent use case diagram generation and behaviour modelling. For the latter, activity diagrams are used to detail the behaviour described in the use case.

Besides that, need for further research emerge. The modular design of the pipeline allows to use of other clustering algorithms, which potentially eliminate current limitations. Extensions to the methodological basis of the pipeline are conceivable. Based on the results obtained by clustering, it could be analysed whether the requirements fully describe the respective use case. This includes preconditions, main and alternative scenarios, as well as postconditions (Pohl & Rupp 2021). A proposal for potential gaps in the use case specification could be a step towards a generative AI in this field.

Acknowledgements

The authors would like to thank AUDI AG for supporting and funding the research project.

References

Ajallouda, L., **Fagroud, F.**, **Zellou, A.** & **Benlahmar, E.** 2022 A systematic literature review of Keyphrases extraction approaches. *International Journal of Interactive Mobile Technologies (IJIM)* **16** (16), 31–58.

Altaf-Ul-Amin, M., **Shinbo, Y.**, **Mihara, K.**, **Kurokawa, K.** & **Kanaya, S.** 2006 Development and implementation of an algorithm for detection of protein complexes in large interaction networks. *BMC Bioinformatics* **7** (1), 207. https://doi.org/10.1186/1471-2105-7-207

Amigó, E., **Gonzalo, J.**, **Artiles, J.** & **Verdejo, F.** 2009 A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Information Retrieval* **12** (4), 461–486.

Bisang, U., **Brünnhäußer, J.**, **Lünnemann, P.**, **Kirsch, L.** & **Lindow, K.** 2022 Evaluate similarity of requirements with multilingual natural language processing. *Proceedings of the Design Society* **2**, 1511–1520.

Cai, J., **Hao, J.**, **Yang, H.**, **Zhao, X.** & **Yang, Y.** 2023 A review on semi-supervised clustering. *Information Sciences* **632**, 164–200.

Casamayor, A., **Godoy, D.** & **Campo, M.** 2012 Functional grouping of natural language requirements for assistance in architectural software design. *Knowledge-Based Systems* **30**, 78–86.

Dalpiaz, F. & **Niu, N.** 2020 Requirements engineering in the days of artificial intelligence. *IEEE Software* **37** (4), 7–10.

Devlin, J., **Chang, M.-W.**, **Lee, K.** & **Toutanova, K.** 2019 BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (No. arXiv:1810.04805). arXiv. https://doi.org/10.48550/arXiv.1810.04805

Dick, J., **Hull, E.** & **Jackson, K.** 2017 *Requirements Engineering*. Springer International Publishing.

Ertel, W. 2017 *Introduction to Artificial Intelligence, Undergraduate Topics in Computer Science*. Springer International Publishing.

Fortunato, S. 2010 Community detection in graphs. *Physics Reports* **486** (3–5), 75–174.

Grootendorst, M. 2020 *KeyBERT: Minimal Keyword Extraction with BERT*. Zenodo.

https://doi.org/10.1017/dsj.2025.10019 Published online by Cambridge University Press

Gülle, K., **Ford, N.**, **Ebel, P.**, **Brokhausen, F.** & **Vogelsang, A.** 2020 Topic modeling on user stories using word mover's distance. In *Proceedings – 7th International Workshop on Artificial Intelligence and Requirements Engineering, AIRE 2020*, IEEE pp. 52–60.

huggingface 2024 Sentence-transformers/all-MiniLM-L6-v2, https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2.

Jurafsky, D. & **Martin, J. H.** 2023 Speech and language processing: An introduction to natural language processing, computational linguistics, and Speech Recognition with Language Models. 3rd https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf

Kleiner, S. & **Kramer, C.** 2013 Model based design with systems engineering based on RFLP using V6. In *Smart Product Engineering*, pp. 93–102. Springer.

Kochbati, T., **Li, S.**, **Gérard, S.** & **Mraidha, C.** 2021 From user stories to models: A machine learning empowered automation, In *Proceedings of the 9th International Conference on Model-Driven Engineering and Software Development* (*Vol.* **10**), pp. 28–40. SCITEPRESS – Science and Technology Publications.

Krog, J., **Sahin, T.** & **Vietor, T.** 2022 Towards a systems engineering methodology for architecture development of vehicle concepts. In *Proceedings of NordDesign 2022*. 12. The Design Society.

Kusner, M., **Sun, Y.**, **Kolkin, N.** & **Weinberger, K.** 2015 From word Embeddings to document distances. In *Proceedings of the 32nd International Conference on Machine Learning*, pp. 957–966. PMLR.

Mikolov, T., **Chen, K.**, **Corrado, G.** & **Dean, J.** 2013 Efficient Estimation of Word Representations in Vector Space (No. arXiv:1301.3781). arXiv. http://arxiv.org/abs/1301.3781

Misra, J., **Sengupta, S.** & **Podder, S.** 2016 Topic cohesion preserving requirements clustering. In *Proceedings – 5th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering, RAISE 2016*, New York, NY, USA: Association for Computing Machinery pp. 22–28.

Mokammel, F., **Coatanéa, E.**, **Coatanéa, J.**, **Nenchev, V.**, **Blanco, E.** & **Pietola, M.** 2018 Automatic requirements extraction, analysis, and graph representation using an approach derived from computational linguistics. *Systems Engineering* **21** (6), 555–575.

Mordecai, Y. & **Dori, D.** 2017 Model-based requirements engineering: Architecting for system requirements with stakeholders in mind. In *2017 IEEE International Systems Engineering Symposium (ISSE)*, pp. 1–8. IEEE.

Pérez-Moure, H., **Lampón, J. F.** & **Cabanelas, P.** 2024 Mobility business models toward a digital tomorrow: Challenges for automotive manufacturers. *Futures* **156**, 103309.

Pohl, K. & **Rupp, C.** 2021 *Basiswissen Requirements Engineering: Aus- Und Weiterbildung Nach IREB-Standard Zum Certified Professional for Requirements Engineering Foundation Level*. dpunkt.verlag.

Reimers, N. & **Gurevych, I.** 2019 Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks (No. arXiv:1908.10084). arXiv. https://doi.org/10.48550/arXiv.1908.10084

Rossetti, G., **Milli, L.** & **Cazabet, R.** 2019 CDLIB: A python library to extract, compare and evaluate communities from complex networks. *Applied Network Science* **4** (1), 1–26.

Salado, A. & **Wach, P.** 2019 Constructing true model-based requirements in SysML. *Systems* **7** (2), 19.

Salman, H., **Hammad, M.**, **Seriai, A.-D.** & **Al-Sbou, A.** 2018 Semantic clustering of functional requirements using agglomerative hierarchical clustering. *Information* **9** (9), 17.

Schaeffer, S. 2007 Graph clustering. *Computer Science Review* **1** (1), 27–64.

https://doi.org/10.1017/dsj.2025.10019 Published online by Cambridge University Press

Schleifer, S., **Lungu, A.**, **Kruse, B.**, **van Putten, S.**, **Goetz, S.** & **Wartzack, S.** 2024 Automatic derivation of use case diagrams from interrelated natural language requirements. In *Proceedings of the 18th International Design Conference* (Vol. **4**), Cambridge University Press (CUP), pp. 2725–2734.

Schopf, T., **Klimek, S.** & **Matthes, F.** 2022 PatternRank: Leveraging pretrained language models and part of speech for unsupervised keyphrase extraction. In *Proceedings of the 14th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2022) - KDIR*; ISBN 978-989-758-614-9; ISSN 2184-3228, SciTePress, 243–248. DOI: 10.5220/0011546600003335

Shchur, O. & **Günnemann, S.** 2019 Overlapping community detection with graph neural networks. In *Deep Learning on Graphs Workshop, KDD*. http://arxiv.org/abs/1909.12201

Sonbol, R., **Rebdawi, G.** & **Ghneim, N.** 2022 The use of NLP-based text representation techniques to support requirement engineering tasks: A systematic mapping review. *IEEE Access* **10**, 62811–62830.

Song, M., **Geng, X.**, **Yao, S.**, **Lu, S.**, **Feng, Y.** & **Jing, L.** 2024 Large language models as zero-shot keyphrase extractors: A preliminary empirical study.

Verein Deutscher Ingenieure, 2021 Development of mechatronic and cyber-physical systems (No. ICS 03.100.40, 31.220.01, 39.020). Beuth.

Walden, D. D., **Roedler, G. J.**, **Forsberg, K.**, **Hamelin, R. D.**, **Shortell, T. M.** & **International Council on Systems Engineering**, eds 2015 *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, 4th ed., Wiley.

Wilking, F., **Horber, D.**, **Schleifer, S.**, **Behringer, M.**, **Miehling, J.**, **Goetz, S.** & **Wartzack, S.** 2024 Utilization of MBSE models for a micro-mobility solution in the context of system of systems. In *2024 19th Annual System of Systems Engineering Conference (SoSE)*, IEEE pp. 24–29.

Zhao, W., **He, Q.**, **Ma, H.** & **Shi, Z.** 2012 Effective semi-supervised document clustering via active learning with instance-level constraints. *Knowledge and Information Systems* **30** (3), 569–587.

https://doi.org/10.1017/dsj.2025.10019 Published online by Cambridge University Press