Technical Report on Neural Language Models and Few-Shot Learning for Systematic Requirements Processing in MDSE

Vincent Bertram, Miriam Boß, Evgeny Kusmenko, Imke Helene Nachmann, Bernhard Rumpe,
Danilo Trotta, Louis Wachtmeister
{bertram,boss,kusmenko,nachmann,rumpe,wachtmeister}@se-rwth.de
RWTH Aachen University, Software Engineering
Aachen, Germany

arXiv:2211.09084v1 [cs.SE] 16 Nov 2022

Abstract

Systems engineering, in particular in the automotive domain, needs to cope with the massively increasing numbers of requirements that arise during the development process. To guarantee a high product quality and make sure that functional safety standards such as ISO26262 are fulfilled, the exploitation of potentials of model-driven systems engineering in the form of automatic analyses, consistency checks, and tracing mechanisms is indispensable. However, the language in which requirements are written, and the tools needed to operate on them, are highly individual and require domain-specific tailoring. This hinders automated processing of requirements as well as the linking of requirements to models. Introducing formal requirement notations in existing projects leads to the challenge of translating masses of requirements and process changes on the one hand and to the necessity of the corresponding training for the requirements engineers.

In this paper, based on the analysis of an open-source set of automotive requirements, we derive domain-specific language constructs helping us to avoid ambiguities in requirements and increase the level of formality. The main contribution is the adoption and evaluation of few-shot learning with large pretrained language models for the automated translation of informal requirements to structured languages such as a requirement DSL. We show that support sets of less than ten translation examples can suffice to few-shot train a language model to incorporate keywords and implement syntactic rules into informal natural language requirements.

1 Introduction

Innovation of modern systems comes from realizing complex functionalities through the interaction of software, electrical and mechanical components. Advanced Driver Assistance Systems (ADASs), autonomous vehicles, and other Cyber-Physical Systems (CPSs) have emerged from this ongoing trend. Due to the heterogeneous and interactive nature of these systems-of-systems, their engineering has become a highly effortful task that faces many challenges [33, 36, 39, 40, 57]. One of them is certainly the need for collaboration of experts from different domains [20]. This challenge also manifests in the rising number of requirements that address stakeholders from heterogeneous domains. In systems engineering, and in particular the automotive domain, requirements are captured as documents that contain text mainly in natural language [41] often with additional information provided through, *e.g.,* pictures or Computer-Aided Design (CAD) models. Experts interpret these textual requirements to enter the design phase, and most often derive details of the implementation directly from them [22]. The ambiguity of natural language, in particular, when interpreted by experts from different backgrounds, as well as the increasing number of requirements may result in decreasing product quality, system failures which are currently detected at late development stages [17] and hinders the implementation of functional safety standards such ISO 26262. Furthermore, the document-based approach to requirements engineering prevents agile development, where automated analyses and syntheses should enable early error detection and fast feedback for the developers.

What is needed are tools to capture, analyze, and process requirements systematically during all phases of the development cycle. An approach to achieve this is Model–Driven Engineering (MDE) [22], which utilizes models as the primary development artifacts. That is, these models serve as documentation and communication basis for engineers, but also as input for analyses and syntheses, such as verification [34], test case [17], or code generation [16]. For instance, MDE can be applied to facilitate the design of Artificial Intelligence (AI)-based systems [2, 26, 37, 38]. Approaches to introduce MDE in the automotive requirements engineering exist [42], but introducing MDE comes with initial costs and efforts for training the domain experts in modeling, and most often, for translating many documents to models [8]. An advantage of using Domain Specific Languages (DSLs) rather than general purpose modeling languages such as the Unified Modeling Language (UML) is that their syntax and semantics [29] can be designed to be intuitive for the model users. As requirements are captured in natural language, we assume that a textual DSL that offers sentence structures and wording that is close to the current

1

Vincent Bertram, Miriam Boß, Evgeny Kusmenko, Imke Helene Nachmann, Bernhard Rumpe, Danilo Trotta, Louis Wachtmeister

formulation of requirements increases intuitiveness of both usage and understanding of models in this DSL significantly. However, in addition to the DSL development costs and the DSL training, once the DSL is developed, the translation of old, unstructured requirements to models in the DSL can be a tremendous effort due to the high number of requirements, requiring time and modeling know-how from the translating developer.

In this paper, we analyze an open-source set of automotive requirements for ADAS and Adaptive Light System (ALS) to understand where formulation inaccuracies occur and how targeted DSL constructs can help eliminate these inaccuracies and increase the level of formality and consistency in these requirements. **The goal and main contribution of this paper is the application and evaluation of few-shot learning of large neural natural language models for the translation of given unstructured requirements to sentences incorporating the new formal DSL constructs.**

Such translation models can be used 1) during the introduction phase of a DSL to automatically translate existing or legacy natural language requirements into the new DSL syntax and 2) to correct natural language inputs in a smart editor when a requirement engineer writes a new requirement as natural text. With this automation supported by the fact that few-shot learning requires only a handful of translation examples to learn a given translation task, **our aim is to facilitate the introduction of highly specialized requirement DSLs, e.g., targeting a single department of a company using specific wording or even a single project.**

The remainder of this paper is structured as follows. Section 2 introduces the technical foundations of our approach; Section 3 highlights the challenges and potentials of an MDE approach for requirements engineering within engineering domains, driven by natural-language text-based documents. Section 4 outlines related work in this area. Section 5 presents an example DSL for capturing requirements in the automotive domain. Section 6 details the automatic translation from natural-language to the DSL. In Section 7 we evaluate the approach in multiple experiments. We discuss threats to validity in Section 8 before Section 9 concludes the paper. This paper is an extended version of the corresponding SLE publication by Bertram et al. 2022 [4].

2 Preliminaries

2.1 Neural Language Models

Our approach for text-to-DSL translation of requirements relies heavily on large transformer-based neural language models. The original transformer architecture [58] is a groundbreaking neural network architecture for sequence-to-sequence processing. It has an encoder-decoder structure,

where the encoder receives a sequence as its input and encodes its content to pass it to the decoder. The decoder iteratively creates the target sequence using a graph search method, e.g., beam search. Instead of using recurrent neurons such as long short-term memorys (LSTMs) the transformer architecture relies exclusively on the attention mechanism to grasp dependencies between sequence elements.

While we are not going to use the original transformer architecture itself, all models employed in this paper are its derivatives. For the automatic translation from natural language requirements to the DSL, we utilize a derivative of Generative Pre-trained Transformer (GPT) [52], which is tailored towards text generation. GPT is a transformer-based decoder-only language model that employs a semi-supervised learning approach [52]. The authors showed that generative unsupervised pretraining on unlabeled data, where, given a sequence of words, the network is supposed to learn to predict the next word with the highest likelihood, and subsequent supervised fine-tuning of the pretrained parameters for a specific downstream task outperformed discriminatively trained models. GPT language models have evolved over the last few years and various variants exist.

In [53], the authors showed that, given a sufficiently large capacity and a large, varied text corpus in training, a language model can solve tasks across different domains for which it has not experienced explicit training, also referred to as zero-shot learning. Furthermore, in [7], the authors show that zero-shot learning is outperformed by few-shot learning. In few-shot learning, the model is given a *support set* consisting of a very small number of training examples demonstrating how to solve a new task as part of the model's input. No weight updates are necessary, i.e. no classical training is performed. The support set is input into the model as part of the query. Based on this context, the language model can then solve the new task for a new input. In contrast to zero-shot learning, few-shot learning enables targeted training for very specific tasks, making it particularly interesting for requirements engineering, a discipline heavily relying on natural language and where training data is often scarce.

For the automatic translation of natural language requirements to a model in the DSL we mainly rely on GPT-J-6B [59, 60], an open-source language model based on the 6.7B GPT-3 [7] network and its hyperparameters. Similarly to GPT-Neo [6], it is also trained on the Pile dataset [25]. According to the authors, its performance is almost on par with the 6.7B GPT-3 network, and it is the best-performing publicly available transformer language model in terms of zero-shot performance on various down-streaming tasks[1].

---

[1]https://arankomatsuzaki.wordpress.com/2021/06/04/gpt-j/

2.2 Datasets

The method presented in this paper is evaluated on a publicly available dataset published by Daimler AG, fostering reproducibility. The dataset stems from the automotive domain and consists of 120 textual requirements [5]. It contains natural language requirements for two typical automotive systems, namely ALS and ADAS.

The requirements of the ALS describe a set of system functions: The functionality that causes the vehicle's direction indicators to flash in response to the steering column lever and hazard warning flasher switch. A function to lower the beams depending on the rotary light switch position and the vehicle setting for daytime running light. An adaptive high beam to control the high beam headlamp depending on the high beam switch and the detection of oncoming vehicles.

For the ADAS system, the dataset contains requirements concerning the main components for adaptive cruise control which maintains the distance to the vehicle in front and a speed set either manually by the driver or via traffic sign detection, provides a distance warning and an emergency brake assistant which reacts to stationary obstacles and to moving obstacles.

The requirements address a variety of typical features of current embedded software-intensive systems, (i.e., software is the key element to realize the functionality) from the automotive domain such as highly distributed and partially safety-critical functionality, hard real-time requirements, a combination of mechanical, mechatronic and electronic components both having reactive and regulating functional behavior and numerous product variants (e.g., due to of different product architectures, legal requirements or special equipment).

3 Model-Driven Requirements Engineering: Challenges and Potentials

As the number of requirements increases with the growing complexity of CPSs, preventing inconsistencies and redundancies has become a highly effortful and error-prone task when performed manually. MDE holds the promise to mend these issues by utilizing modeling languages to formulate requirements in a way that they become comprehensive both for computers and humans within the development process [8, 11, 36]. This section outlines the challenges and potentials of introducing Model-Driven Requirements Engineering (MDRE) in the automotive domain and defines a set of requirements for requirement DSLs to overcome these challenges.

In MDE models are the primary source of information as their mathematical semantics and strict syntax enable a unique understanding of the modeled subject among all stakeholders and computers. Thereby, automated analyses and syntheses performed by the latter become meaningful

and therefore practically useful. However, introducing MDE faces major challenges, which [11] summarizes by stating that the benefits of MDE do not outweigh its costs, which is also the case for MDRE. Part of the arising costs stem from the facts that (1) training engineers to use the modeling languages correctly and efficiently is effortful [31], and (2) transferring the existing documents to models is very time-consuming [22], since modeling itself is an effortful and tedious task [49].

Language engineering means the process of creating suitable DSLs that serve as modeling languages within MDE of a specific domain [13]. Naturally, if this process aims at creating a language that is intuitive for the experts (who do not have a computer-science background), the training effort decreases and with that, the cost of introducing MDE, tackling challenge (1). The contributions of this paper towards tackling these challenges rely on the idea that a textual DSL whose syntax is very close to the phrasing of requirements in natural language used by the experts is intuitive to these experts by construction.

However, engineering such a DSL requires extensive analyses of the natural language used to formulate these requirements, which increases the cost of language engineering significantly [22] and would probably outweigh the cost saving achieved by intuitiveness. Further, given a DSL, automating (parts of) the process to translate natural language requirements to models addresses challenge (2). An implementation of such an automated translation is most likely based on machine learning and Natural Language Processing (NLP) which, to be useful, requires an extensive set of labeled training data. Creating the latter, again, increases the costs of implementing the translation in a way that would probably outweigh its benefits. However, we assume, that the closer the phrasing of the output (models in the DSL) is to the phrasing of the input (requirements in natural language), the more likely it is that few-shot learning approaches suffice for the training. These approaches only need small amounts of labeled data for effective training, and therefore reduce the costs to create a sufficient training dataset. Further, requirements engineering is a very broad field in automotive systems engineering that involves experts from various backgrounds. Engineering a DSL to capture requirements in this domain will be an iterative process during which the language will evolve along with the tools to interpret, analyze and process the models, and this evolution will continue throughout the maintenance phase. A modular design of the language and the underlying tooling is therefore crucial [9, 10].

We derived the following set of requirements on DSLs that facilitate introducing MDRE in the automotive domain by reducing the training and translation efforts:

R1: Since the modelers will not necessarily have a computer-science background, the DSL's syntax shall be based on natural language.

R2: To make the language as intuitive as possible for its users and to enable the application of few-shot learning to implement an automatic translation from natural language requirements to models, the DSL's syntax shall be as close to the phrasing of requirements in natural language used by the modelers as possible.

R3: In the DSL, requirements shall be formulated consistently and with a precise meaning understood by relevant stakeholders, enabling automatic interpretation.

R4: The language and its tooling shall be extensible and maintainable by a modular design.

4 Related Work

4.1 Formalization of Requirements

Requirement DSLs. Trigger action patterns or If-Then constructs as we are going to use them in our DSL are widely used in industry. One example is the graphical formalization language Simplified Universal Pattern [56], which has shown that a large part of the requirements from the automotive domain can be formalized with this concept [3]. An approach that utilizes a DSL for automating type checking of requirements for detecting erroneous syntactic phrasings automatically is proposed in [43]. Therein, the DSL is defined by a context-free grammar and an ontology that defines axiomatic terms and types to allow including typed terms within the requirements. Further tooling includes automatic consistency checking [44]. In [35], and [61] the authors propose approaches to verify the consistency of requirements using temporal logic. Very similar to our DSL, these approaches use a structured English which is defined through a grammar to enable automatic logical analyses on requirements written by non-computer scientists. However, these approaches do not aim at minimizing the difference between the sentence constructs offered by the DSL and the already established natural language used for requirements engineering, which will increase intuitiveness and enable few-shot learning for automated translation.

Structuring Natural Language through Sentence Patterns. In addition to DSLs for requirements development, requirements templates are widely used for standardizing the wording of requirements and increasing their quality. A requirements template provides the requirements engineer with a set of sentence templates as a guide for requirements development. In German-speaking countries, for example, the MASTER template based on [51] is widely used. In addition to the experience-based templates, in the literature there exist approaches that deal with the extraction of requirements templates from legacy requirements. For example, [21] develops a set of sentence templates for quality requirements, [23] presents a metamodel for software requirements. In addition, the PABRE framework presents a method for using requirements templates [24] based on a catalog of 29 QR patterns [54] and 37 non-functional patterns [47]. Because requirements templates provide only a guideline to write requirements, there are no means to detect deviations from the wording proposed by the templates. This has the advantage that requirement engineers and template designers are not limited to formally defined sentence structures when writing requirements. Hence, this approach has the major disadvantage that implementations of automated requirements processing cannot assume correct phrasing of the input.

Formalization by Applying Logic. Approaches that formalize natural language requirements by translations into formal languages that allow interpretations based on logic have been researched for decades, and we will just name a few interesting examples here: The approach presented in [27] translates natural language requirements into a representation in deontic logic which allows to check for consistency. The approach does not utilize machine learning for the translation, but relies on "typographical adjustments, tokenization and morphosyntactic analysis" to extract buckets, *i.e.,* pieces of a specific kind of information, from the sentences. Similarly, [14] propose a technique that enables users to query a conceptual model of the system in natural language and retrieve answers in natural language. The approach targets the verification of natural language requirements given a formal model of the system, *i.e.,* a model in a DSL. Another concept for verifying requirements is proposed in [28] which perceives requirements and a component design each as contracts and verify their correctness by a conjunction of both contracts. The technique relies on modal logic. An overview of the research conducted in formalizing legal requirements is given in [46]. A very recent approach [62] utilizes NLP to transform natural language requirements into a requirements capturing model, which is a semi-formal model. The approach introduces automatic transformations from this representation into metric temporal logic and computation tree logic. All of these approaches take a requirement in natural language as input, and transform it into a structured format that is comparable to a model in a DSL before this model is the input for an automated processing. These approaches, however, do not assure that engineers formulate requirements such that they are uniquely understood among the human stakeholders as well: The intermediate model remains internal to the automated processing, and, being a rephrasing as logical formulas, has an unintuitive format for most human readers.

Representing requirements in a textual DSL structures the sentences such that their understanding among humans

becomes unique and, at the same time, allows the implementation of formalisms to interpret these models appropriately regarding the domain understanding.

4.2 NLP in Requirements Engineering

NoRBERT [30] is a fine-tuned version of BERT [15] for requirements classification. It was trained on the PROMISE NFR dataset [12]. An approach for clustering natural language requirements is presented in [32]. The approach applies clustering to natural language descriptions with the idea of developing a DSL in mind. Apart from requirement classification, machine-learning and NLP can be used for prioritizing requirements [50]. These approaches, however, are not specifically tailored for reducing the *initial* modeling efforts necessary for introducing MDRE.

4.3 Few-Shot Learning for Requirement Classification

There already exist a few approaches concentrating on few-shot learning in requirements engineering. One of these approaches uses transformer models for a named entity-recognition task [45]. Another related approach is a preliminary study on requirement classification using zero-shot learning [1]. The PROMISE NFR dataset is used with pre-trained Transformer models such as BERT and RoBERTa. Since it is in a preliminary status, only a reduced part of the dataset is used. In contrast to our approach, the study is only in a preliminary state and zero-shot learning is used for requirement classification instead of few-shot learning.

Apart from requirement classification, few-shot learning on requirements is also applicable for requirement elicitation as described in [55]. In contrast to other approaches in the area of few-shot learning in requirements engineering, the approach does not take already existing requirements as an input, but chat messages from which requirements for new hidden features are extracted.

4.4 Research Gap and Goal

While neural models have been applied to requirements engineering successfully, in particular for requirements classification, there is a research gap concerning the potential of few-shot learning to tailor language models for specific requirements processing tasks such as reformulation, translation, correction, and the like, where training data is scarce, or where access to computational resources needed to fine-tune very large language models such as the GPT-J-6B is limited. The purpose of this work is hence, to develop a set of tailored rules for systematic requirement definition in a given domain by analyzing existing requirements and to determine if and to which extent few-shot learning can support the translation of legacy requirements into the systematic form, thereby aiming to increase the level of formality in requirements.

5 Example DSL for Structured Requirements

Requirements specify a product under development and thus have a decisive influence on the end product. Missing, incorrect or ambiguously phrased requirements therefore have serious negative effects on the end product and are one of the main causes of additional costs in product development and dissatisfied customers.

The aim of this section is to design an exemplary requirements DSL increasing the degree of formality of requirement documents and hence support automated verification and consistency checks. Instead of designing a general requirement language, our approach is to focus on specific domains or projects and to tailor DSLs accordingly. Furthermore, we employ few-shot learning capabilities of large neural language models in order to transform existing requirements into the new syntax or to support the formulation of new requirements in an editor. In this work, we focus on the automotive domain, and an existing set of ADAS and ALS requirements from [5].

DSLs provide a means to improve the quality of the requirements' phrasing and to facilitate the implementation of tooling to support agile requirements engineering. To engineer a DSL that fulfills the requirements R1-R3 in Section 3, the DSL designer must analyze and understand how the requirements are currently phrased and which meanings are implied by certain formulations.

The DSL is to be used in a natural language domain (*cf.* R1) and must ensure an intuitive readability for the requirements engineer as well as the modeler who is implementing the requirements at a later stage in the development process, *cf.* R3. The developed DSL follows an open-world assumption [18, 19], *i.e.*, whatever is not restricted by the model is allowed. Currently, the DSL focuses on isolated requirements written in one sentence. This is because we are mainly interested in concrete syntax and features such as unambiguity. Complex semantic connections and cross-references will not be discussed here.

A manual analysis of the requirements from [5] reveals a set of ambiguous or unstructured formulations and inconsistencies which might lead to misinterpretations and hence need to be tackled by the DSL approach. The following three ambiguity types are a non-exhaustive selection, which is sufficient as a basis for our few-shot learning experiments.

Ambiguity 1: If-Then Constructs. The If-Then style is an often reoccurring pattern in requirements documents and an often occurring pattern. It reflects the idea that upon the occurrence of a trigger event something must happen. In standard English there are numerous variants of how to express this, making it difficult for requirements engineers

to stick to a consistent scheme, to search for such requirements, and to analyze them automatically. To tackle this difficulty, the first construct we introduce is the If-Then pattern. It not only creates clarity for the different stakeholders, cf. R3, but also makes further processing in trigger action patterns much easier [3]. Therefore, the DSL introduces the two keywords *IF:* and *THEN:*. For this purpose, the individual requirements are divided into two parts, a *trigger* part beginning with the keyword *IF:* and an *action* part starting with the keyword *THEN:*, i.e. a parsing rule is given as `If-Then-Req = 'IF:' trigger 'THEN:' action`. For the sake of simplicity, we assume that the non-terminals `trigger` and `action` can be arbitrary strings.

As an example, the requirement *"When no advancing vehicle is recognized anymore, the high beam illumination is restored within 2 seconds."* is divided into a *trigger* and an *action* part and formulated in the following structure: *"**IF:** no advancing vehicle is recognized anymore, **THEN:** high beam illumination is restored within 2 seconds."*

This way we achieve a partial formalization of the original requirement. The trigger and the action are now clearly separated, and the requirement can be identified as an If-Then requirement easily.

Ambiguity 2: Modal Verbs. The modal verbs (`must, can, should, etc.`) are important for the precise interpretation of requirements [51]. Moreover, in safety-critical systems such as vehicles, there is an important distinction between the available modal verbs. For instance, the modal verb *must* indicates a legal regulation and non-conformity can lead to legal consequences.

Our analysis shows however that the model verb is sometimes skipped. In such cases it might not be clear whether the given sentence is a requirement or a description of an existing system. To enforce the usage of modal verbs we therefore introduce the dedicated keyword *MUST* in the DSL. In cases of missing modal verbs the keyword *MUST* needs to be included at the correct position. If a wrong modal verb such as *"can"* or *"could"* is used, it needs to be exchange by MUST thereby preventing the usage of weak words [51].

An example is the requirement *"Direction blinking: For the USA and Canada, the daytime running light **can** be dimmed by 50 percent during direction blinking on the blinking side."* from the dataset [5]. Here, *"can"* has to be replaced by *"MUST"*. In the DSL, the requirement is modeled as *"Direction blinking: For the USA and Canada, the daytime running light MUST be dimmed by 50 percent during direction blinking on the blinking side.*

Requirements written without a modal verb should also be supplemented appropriately. For example, in the requirement *"The duration of a flashing cycle **is** 1 second."*, the DSL requests a requirement engineer to introduce a *"MUST"* after *"flashing cycle"*. Hence, the model in the DSL would be: *"The duration of a flashing cycle **MUST be** 1 second."*.

Ambiguity 3: Expressions. In requirements, we often need to quantify and compare things. Again, natural language offers many ways to describe comparisons, making it difficult to grasp the information of requirements automatically. For this reason, we introduce a third DSL construct for our DSL, namely logical formulae. Thereby, we are going to allow both words (to keep the language close to natural formulations) and mathematical expressions in the DSL. Depending on the use case, the sentences in the DSL may look different. When using logical signs, these sentences are greatly simplified.

One example is the requirement *"The vehicle's doors are closed automatically when speeding velocity **is bigger than** 10 km/h"* from [5]. Modeled in the DSL integrating logical formulae, the sentence reads as follows: *"The vehicle's doors are closed automatically when speeding velocity **>** 10 km/h"*. When using the alternative construct for logical formulae with descriptive words offered by the DSL, the requirement translates to *"The vehicle's doors are closed automatically when speeding velocity is **GREATER** 10 km/h"*. The difference here is mainly the use of the DSL. Regarding words, a natural language proximity is desired to increase acceptance by the developer (R1,R2). Further keywords include *"LESS"*, *"EQUAL"*, *"LESS OR EQUAL"*, and *"GREATER OR EQUAL"*. Again, such DSL constructs standardize the way how comparisons are formulated in a requirement. This facilitates automated consistency analysis since the operators and their semantics are clearly defined and the variables and constants can be extracted relatively easy from the sentence, e.g. *velocity* and *10 km/h*. Now, if the same variable is used in another requirement, we can 1) link these requirements as treating the same context, e.g. to enable semantic requirement search and 2) are able to check whether the two requirements are consistent.

The three constructs introduced above can and should be combined when appropriate. For instance, the last example needs to be translated to *"IF: speeding velocity is **GREATER** 10 km/h, THEN: the vehicle's doors MUST be closed automatically."*

Now this requirement has almost no degrees of freedom in terms of formulation, and it is easy to extract the trigger variable (speeding velocity), the subject of the action (the vehicle's doors), and the desired state (closed automatically).

The challenge we would like to solve in this paper is how to translate large numbers of legacy requirements into the DSL, how to repair requirements showing syntactical errors, and how to support a requirement engineer writing requirements in the DSL syntax.

6 Translating Unstructured Requirements to DSL

6.1 Overview

To automatically translate legacy natural language requirements into the DSL defined in Section 5, fixing bad formulations and enforcing guidelines usage and regulatory compliance, we utilize transformer-based language models. Since we need to avoid data and resource intensive finetuning (as the necessary amounts of data might lack in project or company-specific design and the required hardware resources might not be accessible/too expensive), we will make use of the few-shot capability, which has been shown to yield good results with only a few training examples. Hence, we rule out BERT and RoBERTa as models for our translation task and stick to the GPT family.

The number of training examples for few-shot learning a new task is usually constrained by the context window, typically allowing 10-100 examples [7]. Sometimes the number of examples is further constrained by the available computational resources. To exploit the available training data as far as possible, we propose a cascaded translation process, where we provide a dedicated few-shot model for each translation task, i.e. trigger-action, modal verbs, and expressions. This reduces the training complexity and yields more focused models as the neural network does not have to decide, which kind of translation(s) to perform but only has to perform the chosen translation it is specialized for.

In case, a single translation needs to be applied, the translation step needs to be preceded either by a manual or an automated classification of the input and the corresponding choice of a translation model. Otherwise a sequential application of multiple models is necessary to incorporate all DSL constructs.

6.2 Requirement Translation

For each translation step, a set of few-shot examples, also referred to as the support set, for the respective requirements category is selected and given to the language model as context. Our hypothesis is that a large capacity language model pretrained on a sufficiently large training set can be taught to solve a specific task such as a reformulation of a given requirement into a systematic form with a very small support set. In contrast to fine-tuning approaches, where a pretrained model is further trained using a downstream task-specific training set, in few-shot learning the weights of the neural network are not adapted. The few-shot examples consist of input/output pairs and are input into the network as a demonstration for the task to be solved, followed by the actual query. The solution to the query is then generated as the model's output based on the support examples from the context.

The translation model can be implemented using any large enough pretrained language model supporting few-shot learning. Models of higher capacity can be expected to perform better in few-shot learned downstream tasks. Based on promising preliminary results compared to other models, we decided to concentrate mainly on GPT-J-6B. GPT Neo [6], another model from the GPT family, showed a lot of variance in its text generation producing a lot of "noisy", unexpected outputs often changing or extending the semantics of the source requirement without an obvious reason in our preliminary experiments.

7 Evaluation

7.1 Research Questions and Metrics

With the experiments conducted for the translation of requirements from unstructured text to DSL our aim was to find answers to the following research questions:

RQ1: Can state-of-the-art language models be employed to translate natural text requirements to systematic formulations based on few-shot learning?

RQ2: How many few-shot learning examples are required to train a translation rule?

To answer RQ1, we applied few-shot learning separately for If-Then requirements, modal verb insertion, and expressions (*cf.* Section 5) (recall that for training each rule we use a separate instance of the language model). As introduced in Section 5 the trained constructs are:

  a) "*'IF:' trigger 'THEN:' action*" syntax for trigger action requirements,
  b) modal verbs enforcement, reframing sentences to contain the "*MUST*" keyword, and
  c) the usage of "*LESS / GREATER / EQUAL / LESS EQUAL / GREATER EQUAL*" operators in constraints.

Input for the few-shot learning were pairs containing the unstructured input and the desired DSL formulation. To evaluate the "trained" language model, it had to transform an unstructured requirement that the model was not given as example into a requirement in the DSL. We then assessed the result of this transformation. Doing so, requires a significant evaluation metric. Metrics such as BLEU [48] typically used to evaluate results of machine translation are not helpful, due to the many different possibilities to express a requirement correctly. Furthermore, semantic correctness does not imply compliance with the expected syntactical structure we would like to enforce. For this reason, we propose a custom evaluation scale with six possible quality classes to assess the correctness of the translated requirements:

Class 1: The translation is both syntactically and semantically correct and fulfills the required formulation rule. No changes required. **Class 2**: The translation is semantically correct, but contains one or two syntactical inaccuracies to

fully implement the desired rule. **Class 3**: Syntactically correct but fails to fully cover the semantics of the source requirement (e.g. by missing a quantifier or a marginal constraint). **Class 4**: The translation contains one or two syntactical inaccuracies to fully implement the desired rule and the semantics is not fully covered, i.e. a combination of 2 and 3. **Class 5**: The translation has grave syntactical errors or does not implement the desired rule. An identity mapping would result in this label, as well (unless the input already implements the desired rule). **Class 6**: The translation is semantically wrong.

A flaw of this scale is that it is not ordinal, e.g. we cannot say that class 2 is better than class 3. However, based on the experience we gathered with it in this work, in most cases a smaller number indicates a more satisfying result.

To answer RQ2 we conducted our evaluation with three differently sized few-shot support sets per translation rule (If-Then, modal verbs, and constraints) consisting of one, four, and six examples each. In case of one-shot learning, *i.e.,* if only one example is presented in training, for the conversion of constraints containing (in)equalities, the result depends on the keyword used in the example. For this reason, we tried two different one-shot trainings, *i.e.,* for introducing *"EQUAL"*, and *"LESS OR EQUAL"*. The requirements used for testing were **not** present in the support sets.

7.2 Concrete Training Examples

In this section we provide excerpts of the training and prediction experiments to give the reader an intuition on how the GPT-J-6B model dealt with the translation, which constructs were particularly challenging, and how the number of examples affected the quality of the translation. The full support sets and the complete translation experiment results can be found in the appendix in Table A.1-Table A.22.

If-Then, 1 training example: To one-shot train the translation of a requirement to the desired If-Then syntax we use the following input and output pair:

"Input: If a defective illuminant is detected, the information about the defective illuminant is transmitted to the instrument cluster.
DSL: IF: defective illuminant is detected, THEN: information about the defective illuminant is transmitted to the instrument cluster."

As we can see in the example, the two keywords *"IF:"* and *"THEN:"* are included in the target sentence. Apart from that the sentence remains almost unchanged, making it relatively easy for the model to learn the rule.

Having seen this example, the model is already able to apply the rule perfectly to some examples of the test set, yielding a class 1 rating according to the scheme given above. For instance, the requirement

"If tip-blinking was activated shortly before deactivation of the hazard warning, this is not considered during the deactivation of the hazard warning."

is correctly translated to

"IF: tip-blinking was activated shortly before deactivation of the hazard warning, THEN: this is not considered during the deactivation of the hazard warning ".

However, some other examples are translated incompletely or wrong, e.g. the requirement

"With activated darkness switch (only armored vehicles) the cornering light is not activated."

is translated to

"IF: darkness switch is activated, THEN: cornering light is not activated. "

While the keywords are included at the correct position, the model drops the information in parentheses stating that the requirement only applies to armored vehicles. For this reason, we consider this translation as syntactically correct, but semantically incomplete, resulting in a class 3 rating.

At this point the model has not yet fully learnt to implement the If-Then syntax if the sentence does not start with an *"if"*. For instance, the example

"When moving the pitman arm in position turn left the vehicle flashes all left direction indicators (front left, exterior mirror left, rear left) synchronically with pulse ratio bright to dark 1:1."

remains unchanged by the one-shot model. This yields a class 5 rating, as the semantics is correct, but the desired syntax is not implemented. Interestingly, another requirement starting with *"when"* (*"When hazard warning is deactivated and the pit arm is in position direction blinking left or direction blinking right, the direction blinking cycle should be released. "*) results in the output

"WHEN: hazard warning is deactivated, THEN: direction blinking cycle should be released ".

Here the model introduces a new keyword (*"WHEN:"*), which did not happen in the previous example. Furthermore the second condition concerning the pit arm position has been dropped, resulting in a class 6 rating due to both syntactical and semantical errors.

If-Then, 4 training examples: Aiming to improve the translation quality, we introduce three additional training example pairs:

"Input: If no advancing vehicle is recognized any more, the high beam illumination is restored within 2 seconds.
DSL: IF: no advancing vehicle is recognized any more, THEN: high beam illumination is restored within 2 seconds.
###
Input: If the light rotary switch is in position "auto", the adaptive high beam headlights are activated by moving the pitman arm to the back.
DSL: IF: the light rotary switch is in position "auto" and the pitman arm is moved back, THEN: the adaptive high beam headlights are activated.
###
Input: (a) When the driver enables the cruise control (by pulling the cruise control lever or by pressing the cruise control lever

up or down), the vehicle maintains the set speed if possible.
DSL: IF: driver enables the cruise control by pulling the cruise control lever or by pressing the cruise control lever up or down, THEN: the vehicle maintains the set speed if possible. "

The three additional examples lead to an improvement in translation quality, for instance, instead of generating a new keyword *"WHEN:"*, the model now knows due to the last training example that it has to replace *"When"* by *"IF:"*. So now, the hazard warning requirement is translated correctly to

"IF: hazard warning is deactivated and the pit arm is in position direction blinking left or direction blinking right, THEN: the direction blinking cycle should be released." and yields a class 1 rating.

If-Then, 6 training examples: To further improve the translation quality, the following examples have been added:
" Input: If the driver pushes down the cruise control lever with cruise control activated up to the first resistance level, the speed set point of the cruise control is reduced by N.
DSL: IF: driver pushes down the cruise control lever with cruise control activated up to the first resistance level, THEN: the speed set point of the cruise control is reduced by N.
###
Input: By pushing the brake or the hand brake, the cruise control is deactivated until it is activated again.
DSL: IF: the brake or the hand brake is pushed, THEN: the cruise control is deactivated until it is activated again."

In this variant the translation of the armored vehicle requirement is improved to a class 2 rating yielding the text *"IF: with activated darkness switch (only armored vehicles), THEN: the cornering light is not activated. "* Here the model manages to keep the information on armored vehicles in parentheses, but chooses a strange formulation in the if-part, i.e. small syntactic fixes are required. Hence, the model improves from a class 3 to a class 2 grade.

Requirements of the form *"Context: requirement text."* were particularly problematic for the model. The context was often interpreted as a condition. For instance, the requirement *"Distance Warning: The vehicle warns the driver visually and/or acoustically if the vehicle is closer to the car ahead than allowed by the safety distance."* is translated to *"IF: distance warning is activated, THEN: the vehicle warns the driver visually and/or acoustically"*. This problem persists for all sizes of support sets we have tried, yielding a class 6 grading. Probably, the problem also persists for other deviating sentence structures.

Modal verb, 1 training examples: The one-shot support set for the insertion of modal verbs is given as:
"Input: The maximum deviation of the pulse ratio should be below the cognitive threshold of a human observer.
DSL: The maximum deviation of the pulse ratio MUST be below the cognitive threshold of a human observer."

Given this single example the model is able to insert the *"MUST"* keyword into simple sentences. For instance, the descriptive sentence *"The frame rate of the camera is 60 Hz."* is translated to a real requirement *"The frame rate of the camera MUST be 60 Hz."*, thereby yielding a class 1 grading. However, at this point it fails to insert the *"MUST"* keyword in most other test examples.

Modal verb, 4 training examples:" For the 4-shot support set we use the following additional training example pairs:
"Input: Direction blinking: For USA and CANADA, the daytime running light shall be dimmed by 50% during direction blinking on the blinking side.
DSL: Direction blinking: For USA and CANADA, the daytime running light MUST be dimmed by 50% during direction blinking on the blinking side.
###
Input: The adaptation of the pulse ratio has to occur at the latest after two complete flashing cycles.
DSL: The adaptation of the pulse ratio MUST occur at the latest after two complete flashing cycles.
###
Input: The duration of a flashing cycle is 1 second.
DSL: The duration of a flashing cycle MUST be 1 second."

This helps the model to translate more inputs correctly, e.g. while the requirement *"A flashing cycle (bright to dark) has to be always completed, before a new flashing cycle can occur"* remained unchanged by the model in the one-shot case, it is now translated to *"A flashing cycle (bright to dark) MUST be always completed, before a new flashing cycle can occur."* This yields a class 1 grade.

Similarly, the example *"Also after 1000 flashing cycles the cumulated deviation will not exceed 0.05s."* is now translated to textit*"Also after 1000 flashing cycles the cumulated deviation MUST NOT exceed 0.05s."*, again yielding class 1.

Modal verb, 6 training examples: Adding the following additional two training examples to the support set further improves the translation quality:
"Input: A flashing cycle (bright to dark) will always be completed, before a new flashing cycle can occur.
DSL: A flashing cycle (bright to dark) MUST always be completed, before a new flashing cycle can occur.
###
Translate input to DSL Input: With subvoltage the ambient light is not available.
DSL: With subvoltage the ambient light MUST not be available."

The two inputs *"With subvoltage the ambient light is not available."* and *"Low beam illuminant shall be LED."* are now correctly translated to *"With subvoltage the ambient light MUST not be available."* and *"Low beam illuminant MUST be LED.".*

Some inputs are still not translated correctly, e.g. *"The functions of the system are classified as safety relevant in with*

respect to ISO 26262." But also the seemingly simpler requirement formulation *"The vehicle does not exceed a set speed.".*

Expressions with keywords, 1 training examples: In the next experiment, we aim to detect expressions, in particular comparisons in requirements and insert keywords such as *"EQUALS"* or *"GREATER"* into the sentences. The one-shot support set is given as:

"Input: The maximum deviation of the pulse ratio should be below the cognitive threshold of a human observer.
DSL: The maximum deviation of the pulse ratio should be LESS OR EQUAL cognitive threshold of human observer."

The problem here is the relatively large number of keywords. The model cannot infer all the keywords from a single example. Nevertheless, it tries to apply the *"LESS OR EQUAL"* keyword to some test inputs at the correct place. For instance, the requirement
"The luminous intensity of the daytime running light must be lower than 400cd." is transformed to
"The luminous intensity of the daytime running light must be LESS OR EQUAL 400cd.".
The result has a semantic flaw as it allows an equality which is not the case for the original, hence resulting in a class 3 rating.

Interestingly, the model transforms *"The maximum curb weight of the vehicle must be no more than 3.5t."* to *"The maximum curb weight of the vehicle must be LESS THAN 3.5t.",* thereby introducing a keyword it has not seen in training. In contrary to the previous requirement, equality is allowed here in the original but is excluded by the mode, again resulting in a lass 3 rating.

We tried a one shot variant with an exampling using the *"EQUAL"* keyword instead of *"LESS OR EQUAL"*. As expected, the model then only used this keyword and hence, performed better where *"EQUAL"* was required and worse otherwise.

Expressions with keywords, 4 training examples: In the next experiment we used the four following examples as support set:

"Input: The duration of a flashing cycle is 1 second DSL: The Duration of a flashing cycle is EQUAL 1 second.
###
Input: The maximum deviation of the pulse ratio should be below the cognitive threshold of a human observer.
DSL: The maximum deviation of the pulse ratio should be LESS OR EQUAL cognitive threshold of human observer.
###
Input: The minimal number of seatbelts used has to be 1.
DSL: The number of seatbelts used has to be GREATER OR EQUAL 1.
###
Input: The vehicles doors are closed automaticly when speeding velocity is bigger than 10km/h.
DSL: The vehicles doors are closed automaticly when speeding velocity is GREATER 10km/h."

Surprisingly, the model was biased towards inserting the *"LESS THAN"* keyword, which was not present in the support set at all. The overall improvements were minor. The luminous intensity requirement was now transformed correctly to *"The luminous intensity of the daytime running light must be LESS THAN 400cd.",* yielding a class 1 rating as the only requirement.

Experessions with keywords, 6 training examples: For the last experiment we added the following two training examples: *"Input: Also after 1000 flashing cycles the cumulated deviation must not exceed 0.05s. DSL: After 1000 flashing cycles the cumulated Deviation must be GREATER OR EQUAL 0.05s.*
###
Input: The cruising speed has to be set at a speed which exceeds 10km/h. DSL: The cruising speed has to be GREATER 10km/h."

In this experiment additionally the maximum curb weight requirement was transformed perfectly to *" The maximum curb weight of the vehicle must be LESS OR EQUAL 3.5t.",* now including the equality part.

The experiment shows a main drawback inherent to models such as GPT. The models use statistically learned sequences without understanding the semantics. For instance, the model often confuses less-than and greater-than inequalities, which are very similar from the syntactical point of view, but have a completely different semantics. For instance, the model delivers translations such as *"The minimum distance to a vehicle in front has to be LESS OR EQUAL 5m.".* While it is obvious to a human reader that *"GREATER"* should be used here, a neural language model has no logical interpretation.

Another problem which becomes evident in this evaluation is that the language model has difficulties reformulating sentences when a pure insertion of a keyword is insufficient. For instance, the example *"The vehicle's horn must not be louder than 50dB."* is transformed to *"The vehicle's horn must be LESS OR EQUAL 50dB.".* While the keyword used is correct, the sentence has a semantic flaw: it is not the horn that has to be less or equal 50dB but its intensity, i.e. the perfect translation would be *"'The intensity of the vehicle's horn must be LESS OR EQUAL 50dB.".* The model struggles to insert the physical quantity (intensity) here.

Expression Extraction Additionally, we performed experiments aiming to extract mathematical formulas only containing the key operators and variable names. For instance, the input *"The braking distance can not be longer than 300m."* yields the output *"braking distance <= 300m".* Such an extraction can be useful to implement tracing and consistency checking of multiple requirements. For instance, if the variable braking distance is found in multiple requirements and featuring different bounds, this can be found automatically and shown to a human expert to check whether the requirements are indeed contradictory. Further successful extractions we generated using the model are *"horn loudness <= 50dB",* *"low beam illuminant = LED",* and *"distance to vehicle*

in front >= 5m". Most others showed slight to medium inacuracies, e.g. *"maximum velocity <= 260km/h"* or *"blinking lights = 3s"*.

7.3 Evaluation Summary

An overview of all experiment results is summarized in Table 1. As expected, in each of the three experiments, the translation quality improved with larger support sets. It is fascinating however, how steep the learning curve is. It suggests that few-shot learning can deal with natural language processing (NLP) tasks in requirements engineering even when only small training sets are available.

In the first experiment (If-Then translation), given a support set of six training examples, six out of eleven examples achieve a class 1 rating, *i.e.,* the model did a perfect translation. Further, the results with only four training pairs in the support set were still good. However, two requirements were transformed incorrectly. An example for a class 1 translation is: *"Emergency Brake Assist: The vehicle decelerates in critical situations to a full standstill."* is transformed to *"IF: emergency brake assist is activated, THEN: the vehicle decelerates in critical situations to a full standstill."* Here the malformed sentence is restructured with the verb "is activated", separating trigger and action. Furthermore, the model learns to replace words such as "if", "once", "when" by the keyword "IF:" and to insert "THEN:" at the correct position. The two requirements which were not transformed correctly even for six training examples both contained two conditions and were hence more difficult to translate. Apparently, the network has difficulties processing inputs consisting of several sentences, e.g. multiple trigger-action clauses. However, subdividing such requirements into separate inputs seems to solve the problem.

The second experiment (modal verbs translation) was probably the easiest, which is not surprising. The language model only needs to add or replace the right verb with "MUST". For instance "The frame rate of the camera is 60 Hz." was successfully translated to "The frame rate of the camera MUST be 60 Hz". The model did this well for all but two requirements given six training examples. One of the two outliers contained a negation, which was not explicitly trained.

For the third experiment, dealing with constraints (introducing (in-)equalities), to explore the capability of our model to identify and generate constraint sentences, we crafted an additional dataset consisting of six few-shot training and six test examples featuring equality and inequality clauses. Due to more variance in the desired translations (five keywords to choose from) and the difficulty encountered by the model to transform adjectives to subjects (e.g. "the horn has to be louder than" needs to become "the horn intensity has to be GREATER than"), we added three more training examples focusing on this issue. It did however not lead to the desired results, the model did not learn to transform the sentence accordingly. Sentences for which no such adjective-to-subject

transformation was necessary had a better success rate in the test set. For instance, "The maximum curb weight of the vehicle must be no more than 3.5t." was successfully translated to "The maximum curb weight of the vehicle must be LESS OR EQUAL 3.5t." Some translations featured a wrong operator, but were otherwise syntactically correct. In applications such as smart editors, this could still be a helpful way to sensitize the requirement engineers for the correct syntax.

We conducted a variant of the third experiment, where a) the natural language operators were replaced by mathematical ones, i.e. $<, \leq, >, \geq, =$, and b) only the variables/constants of the (in-)equality were extracted in the support set, i.e. without keeping the rest of the sentence. The model managed to perform these transformations surprisingly well (except for one example where the model put a "$\leq$" instead of "$=$" for the maximum velocity), e.g. from "The vehicles horn must not be louder than 50dB" the model extracted "horn loudness $\leq$ 50dB". Such extractions can be used in tracing, e.g. to identify assignments or conditions in code which are inconsistent with the underlying requirements. Variables and their values or bounds can be extracted by a language model and looked up in code automatically.

8 Threats to Validity

Construction Validity. An objective metric of requirement classification and translation is not feasible per se and depends on the context, perspective of the reader, and other factors. For our evaluation we let five software engineers label the translations independently and took the majority vote after a discussion to mitigate the risk of bias.

Internal Validity. Our evaluation dataset might have a too small variance: The legacy examples stem from a single project and are mostly well written, there are no completely ill-formed examples. This might render the translation process too easy for the language model.

External Validity. Generalizability is a major concern in requirements engineering as findings and algorithms need to be transferred to unseen projects and domains. The methodology presented in this paper is designed for domain-specific adaptations, *i.e.,* the results are only valid for the datasets at hand. For the translation task, we cannot rule out that different or more complex syntactic elements required in other domains cannot be realized using few-shot learning or require larger support sets. Generalizability of the concrete few-shot models derived in this paper across different domains is probably not feasible due to differences in domain-specific wordings and formulation approaches, but also not pursued in this work since each domain should use tailored support sets.

Table 1. Evaluation results for the translation experiments from natural language requirements to domain-specific syntax.

# of Training Set	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	total
Translation results for If-Then structure using GPT-J6B							
1	2	0	1	0	6	2	11
4	5	2	0	1	1	2	11
6	6	3	0	0	0	2	11
Translation results for modal verbs structure using GPT-J6B							
1	3	0	0	0	5	0	8
4	5	0	0	0	3	0	8
6	6	0	0	0	2	0	8
Translation results for propositional logic Structure using GPT-J6B							
1 (trained on keyword: equal)	2	0	1	0	4	1	8
1 (trained on keyword: less or equal)	0	0	4	0	4	0	8
4	1	0	1	0	3	3	8
6	2	0	3	0	2	1	8
8	3	0	0	0	3	2	8

9 Conclusion

In this paper, we have shown how neural language models such as representatives of the GPT family can support requirements engineering without the need for resource and data intensive fine-tuning. Our most important result is that few-shot learning of language models can be applied to translate legacy requirements into a given structured DSL form automatically. However, language models available today still require human supervision. Since context space available in a language model for few-shot learning is limited to a very small number of examples, we suggest to manually or automatically classify a requirement before translation to choose the few-shot examples to be used for a given translation task.

We draw the conclusion that few-shot learning is a powerful tool for the processing of natural language requirements which does not require costly training from scratch and expensive hardware and is hence accessible even for small companies with limited resources or when training data is scarce. It has the potential to accelerate the formalization of legacy requirements and improve the resulting requirements quality and syntactical consistency. However, as of now, supervision cannot be eliminated, binding human resources but limiting the task to correction.

Instead of having to rely on the generalizability of a single language model for all projects, the few-shot learning approach can be employed to tailor language models to new projects and application domains quickly. In very specific areas, singular fine-tunings to the domain might be necessary, e.g. we realized in our experiments that in some cases the language models had difficulties processing automotive-related language terms correctly.

We believe that the presented technology can be used for a variety of tasks in model-driven requirements engineering including but not limited to the translation of legacy requirements, autocompletion for smart editors, requirement lookup and information extraction. While there is still room for improvement, it is reasonable to expect that the capacity of language models will steadily grow larger in the next years providing a continuously increasing few-shot learning performance.

References

[1] Waad Alhoshan, Liping Zhao, Alessio Ferrari, and Keletso J. Letsholo. 2022. A Zero-Shot Learning Approach to Classifying Requirements: A Preliminary Study. In *Requirements Engineering: Foundation for Software Quality*, Vincenzo Gervasi and Andreas Vogelsang (Eds.). Springer International Publishing, Cham, 52–59.

[2] Abdallah Atouani, Jörg Christian Kirchhof, Evgeny Kusmenko, and Bernhard Rumpe. 2021. Artifact and Reference Models for Generative Machine Learning Frameworks and Build Systems. In *Proceedings of the 20th ACM SIGPLAN International Conference on Generative Programming: Concepts and Experiences (GPCE 21)*, Eli Tilevich and Coen De Roover (Eds.). ACM SIGPLAN, 55–68.

[3] Jan Steffen Becker, Vincent Bertram, Tom Bienmüller, Udo Brockmeyer, Heiko Dörr, Thomas Peikenkamp, and Tino Teige. 2018. Interoperable Toolchain for Requirements-Driven Model-Based Development. In *9th European Congress Embedded Real Time Software and Systems ERTS '2018*.

[4] Vincent Bertram, Miriam Boß, Evgeny Kusmenko, Imke Nachmann, Bernhard Rumpe, Danilo Trotta, and Louis Wachtmeister. 2022. Neural Language Models and Few-Shot Learning for Systematic Requirements Processing in MDSE. In *International Conference on Software Language Engineering (SLE'22)* (Auckland, New Zealand). ACM. https://doi.org/10.1145/3567512.3567534

[5] Vincent Bertram, Shahar Maoz, Jan Oliver Ringert, Bernhard Rumpe, and Michael von Wenckstern. 2017. Component and Connector Views in Practice: An Experience Report. In *Conference on Model Driven Engineering Languages and Systems (MODELS'17)* (Austin). IEEE, 167–177.

[6] Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. 2021. *GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow.* https://doi.org/10.5281/zenodo.5297715

[7] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam,

Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165* (2020).

[8] Antonio Buchhiarone, Jordi Cabot, Richard F. Paige, and Alfonso Pierantonio. 2020. Grand Challenges in Model-Driven Engineering: An Analysis of the State of Research. *Software and Systems Modeling* 19, 1 (2020).

[9] Arvid Butting, Robert Eikermann, Katrin Hölldobler, Nico Jansen, Bernhard Rumpe, and Andreas Wortmann. 2020. A Library of Literals, Expressions, Types, and Statements for Compositional Language Design. *Special Issue dedicated to Martin Gogolla on his 65th Birthday, Journal of Object Technology* 19, 3 (October 2020), 3:1–16. Special Issue dedicated to Martin Gogolla on his 65th Birthday.

[10] Arvid Butting, Jerome Pfeiffer, Bernhard Rumpe, and Andreas Wortmann. 2020. A Compositional Framework for Systematic Modeling Language Reuse. In *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*. ACM, 35–46.

[11] Jordi Cabot, Robert Clarisó, Marco Brambilla, and Sébastien Gérard. 2018. Cognifying Model-Driven Software Engineering. In *Software Technologies: Applications and Foundations*.

[12] Jane Cleland-Huang, Sepideh Mazrouee, Huang Liguo, and Dan Port. 2007. *nfr [Data set]*. https://doi.org/10.5281/zenodo.268542

[13] Benoit Combemale, Robert France, Jean-Marc Jézéquel, Bernhard Rumpe, James Steel, and Didier Vojtisek. 2016. *Engineering Modeling Languages: Turning Domain Knowledge into Tools*. Chapman & Hall/CRC Innovations in Software Engineering and Software Development Series.

[14] Hercules Dalianis. 1992. A method for validating a conceptual model by natural language discourse generation. In *Advanced Information Systems Engineering*.

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[16] Imke Drave, Akradii Gerasimov, Judith Michael, Lukas Netz, Bernhard Rumpe, and Simon Varga. 2021. A Methodology for Retrofitting Generative Aspects in Existing Applications. *Journal of Object Technology* 20 (November 2021), 1–24. https://doi.org/10.5381/jot.2021.20.2.a7

[17] Imke Drave, Timo Greifenberg, Steffen Hillemacher, Stefan Kriebel, Evgeny Kusmenko, Matthias Markthaler, Philipp Orth, Karin Samira Salman, Johannes Richenhagen, Bernhard Rumpe, Christoph Schulze, Michael Wenckstern, and Andreas Wortmann. 2019. SMArDT modeling for automotive software testing. *Software: Practice and Experience* 49, 2 (February 2019), 301–328.

[18] Imke Drave, Timo Henrich, Katrin Hölldobler, Oliver Kautz, Judith Michael, and Bernhard Rumpe. 2020. Modellierung, Verifikation und Synthese von validen Planungszuständen für Fernsehausstrahlungen. In *Modellierung 2020* (Vienna), Dominik Bork, Dimitris Karagiannis, and Heinrich C. Mayr (Eds.). Gesellschaft für Informatik e.V., 173–188.

[19] Imke Drave, Oliver Kautz, Judith Michael, and Bernhard Rumpe. 2019. Semantic Evolution Analysis of Feature Models. In *International Systems and Software Product Line Conference (SPLC'19)* (Paris), Thorsten Berger, Philippe Collet, Laurence Duchien, Thomas Fogdal, Patrick Heymans, Timo Kehrer, Jabier Martinez, Raúl Mazo, Leticia Montalvillo, Camille Salinesi, Xhevahire Tërnava, Thomas Thüm, and Tewfik Ziadi (Eds.). ACM, 245–255.

[20] Imke Drave, Bernhard Rumpe, Andreas Wortmann, Joerg Berroth, Gregor Hoepfner, Georg Jacobs, Kathrin Spuetz, Thilo Zerwas, Christian Guist, and Jens Kohl. 2020. Modeling Mechanical Functional Architectures in SysML. In *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*. ACM, 79–89.

[21] Jonas Eckhardt, Andreas Vogelsang, and Henning Femmer. 2016. An Approach for Creating Sentence Patterns for Quality Requirements.

In *2016 IEEE 24th International Requirements Engineering Conference Workshops (REW)*. 308–315.

[22] Robert France and Bernhard Rumpe. 2007. Model-driven Development of Complex Software: A Research Roadmap. *Future of Software Engineering (FOSE '07)* (May 2007), 37–54.

[23] Xavier Franch, Cristina Palomares, Carme Quer, Samuel Renault, and François Lazzer. 2010. A Metamodel for Software Requirement Patterns. In *Requirements Engineering: Foundation for Software Quality*.

[24] Xavier Franch, Carme Quer, Samuel Renault, Cindy Guerlain, and Cristina Palomares. 2013. *Constructing and Using Software Requirement Patterns*.

[25] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. 2020. The Pile: An 800GB Dataset of Diverse Text for Language Modeling. *arXiv preprint arXiv:2101.00027* (2020).

[26] Nicola Gatto, Evgeny Kusmenko, and Bernhard Rumpe. 2019. Modeling Deep Reinforcement Learning Based Architectures for Cyber-Physical Systems. In *Proceedings of MODELS 2019. Workshop MDE Intelligence* (Munich), Loli Burgueño, Alexander Pretschner, Sebastian Voss, Michel Chaudron, Jörg Kienzle, Markus Völter, Sébastien Gérard, Mansooreh Zahedi, Erwan Bousse, Arend Rensink, Fiona Polack, Gregor Engels, and Gerti Kappel (Eds.). 196–202.

[27] Vincenzo Gervasi and Didar Zowghi. 2005. Reasoning about Inconsistencies in Natural Language Requirements. *ACM Trans. Softw. Eng. Methodol.* 14, 3 (2005).

[28] Gregor Goessler and Jean-Baptiste Raclet. 2009. Modal Contracts for Component-Based Design. In *2009 Seventh IEEE International Conference on Software Engineering and Formal Methods*.

[29] David Harel and Bernhard Rumpe. 2004. Meaningful Modeling: What's the Semantics of "Semantics"? *IEEE Computer* 37, 10 (October 2004), 64–72.

[30] Tobias Hey, Jan Keim, Anne Koziolek, and Walter F. Tichy. 2020. NoRBERT: Transfer learning for requirements classification. In *2020 IEEE 28th International Requirements Engineering Conference (RE)*. IEEE, 169–179.

[31] John Hutchinson, Jon Whittle, and Mark Rouncefield. 2014. Model-driven engineering practices in industry: Social, organizational and managerial factors that lead to success or failure. *Science of Computer Programming* 89 (2014).

[32] Katharina Juhnke, Alexander Nikic, and Matthias Tichy. 2021. Clustering Natural Language Test Case Instructions as Input for Deriving Automotive Testing DSLs. *The Journal of Object Technology* 20, 3 (2021).

[33] Nils Kaminski, Evgeny Kusmenko, and Bernhard Rumpe. 2019. Modeling Dynamic Architectures of Self-Adaptive Cooperative Systems. *The Journal of Object Technology* 18, 2 (July 2019), 1–20. https://doi.org/10.5381/jot.2019.18.2.a2 The 15th European Conference on Modelling Foundations and Applications.

[34] Hendrik Kausch, Mathias Pfeiffer, Deni Raco, and Bernhard Rumpe. 2020. An Approach for Logic-based Knowledge Representation and Automated Reasoning over Underspecification and Refinement in Safety-Critical Cyber-Physical Systems. In *Combined Proceedings of the Workshops at Software Engineering 2020* (Innsbruck), Regina Hebig and Robert Heinrich (Eds.), Vol. 2581. CEUR Workshop Proceedings. http://ceur-ws.org/Vol-2581/

[35] S. Konrad and B.H.C. Cheng. 2005. Real-time specification patterns. In *Proceedings. 27th International Conference on Software Engineering, 2005. ICSE 2005.*

[36] Stefan Kriebel, Evgeny Kusmenko, Bernhard Rumpe, and Michael von Wenckstern. 2018. Finding Inconsistencies in Design Models and Requirements by Applying the SMARDT Process. In *Tagungsband des Dagstuhl-Workshop MBEES: Modellbasierte Entwicklung eingebetteter Systeme XIV (MBEES'18)*. Univ. Hamburg.

[37] Evgeny Kusmenko, Sebastian Nickels, Svetlana Pavlitskaya, Bernhard Rumpe, and Thomas Timmermanns. 2019. Modeling and Training of Neural Processing Systems. In *Conference on Model Driven Engineering Languages and Systems (MODELS'19)* (Munich), Marouane Kessentini, Tao Yue, Alexander Pretschner, Sebastian Voss, and Loli Burgueño (Eds.). IEEE, 283–293.

[38] Evgeny Kusmenko, Svetlana Pavlitskaya, Bernhard Rumpe, and Sebastian Stüber. 2019. On the Engineering of AI-Powered Systems. In *ASE19. Software Engineering Intelligence Workshop (SEI19)* (San Diego, California, USA), Lisa O'Conner (Ed.). IEEE, 126–133.

[39] Evgeny Kusmenko, Alexander Roth, Bernhard Rumpe, and Michael von Wenckstern. 2017. Modeling Architectures of Cyber-Physical Systems. In *European Conference on Modelling Foundations and Applications (ECMFA'17)* (Marburg) *(LNCS 10376)*. Springer, 34–50.

[40] Evgeny Kusmenko, Bernhard Rumpe, Sascha Schneiders, and Michael von Wenckstern. 2018. Highly-Optimizing and Multi-Target Compiler for Embedded System Models: C++ Compiler Toolchain for the Component and Connector Language EmbeddedMontiArc. In *Conference on Model Driven Engineering Languages and Systems (MODELS'18)* (Copenhagen). ACM, 447 – 457.

[41] Grischa Liebel, Matthias Tichy, and Eric Knauss. 2019. Use, potential, and showstoppers of models in automotive requirements engineering. (2019).

[42] Grzegorz Loniewski, Emilio Insfran, and Silvia Abrahão. 2010. A Systematic Review of the Use of Requirements Engineering Techniques in Model-Driven Development. In *Model Driven Engineering Languages and Systems. MODELS'2010*.

[43] Nesredin Mahmud, Cristina Seceleanu, and Oscar Ljungkrantz. 2015. ReSA: An ontology-based requirement specification language tailored to automotive systems. In *10th IEEE International Symposium on Industrial Embedded Systems (SIES)*.

[44] Nesredin Mahmud, Cristina Seceleanu, and Oscar Ljungkrantz. 2016. ReSA Tool: Structured requirements specification and SAT-based consistency-checking. In *2016 Federated Conference on Computer Science and Information Systems (FedCSIS)*.

[45] Garima Malik, Mucahit Cevik, Swayami Bera, Savas Yildirim, Devang Parikh, and Ayse Basar. 2022. Software Requirement Specific Entity Extraction using Transformer Models. *Proceedings of the Canadian Conference on Artificial Intelligence* (2022).

[46] Paul N. Otto and Annie I. Anton. 2007. Addressing Legal Requirements in Requirements Engineering. In *15th IEEE International Requirements Engineering Conference (RE 2007)*.

[47] Cristina Palomares, Carme Quer, Xavier Franch, Cindy Guerlain, and Samuel Renault. 2012. A catalogue of non-technical Requirement Patterns. In *2012 Second IEEE International Workshop on Requirements Patterns (RePa)*.

[48] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*. 311–318.

[49] Tanumoy Pati, Sowmya Kolli, and James H. Hill. 2017. Proactive modeling: a new model intelligence technique. *Software & Systems Modeling* 16 (2017).

[50] Anna Perini, Angelo Susi, and Paolo Avesani. 2013. A Machine Learning Approach to Software Requirements Prioritization. *IEEE Transactions on Software Engineering* 39, 4 (2013).

[51] Klaus Pohl and Chris Rupp. 2021. *Basiswissen requirements engineering: Aus-und Weiterbildung nach IREB-Standard zum certified professional for requirements engineering foundation level*.

[52] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. (2018).

[53] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multi-task learners. *OpenAI blog* 1, 8 (2019), 9.

[54] Samuel Renault, Oscar Mendez-Bonilla, Xavier Franch, and Carme Quer. 2009. A pattern-based method for building requirements documents in call-for-tender processes. *International Journal of Computer Science and Applications* 6 (2009).

[55] Lin Shi, Mingzhe Xing, Mingyang Li, Yawen Wang, Shoubin Li, and Qing Wang. 2020. Detection of Hidden Feature Requests from Massive Chat Messages via Deep Siamese Network. In *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*. 641–653.

[56] Tino Teige, Tom Bienmüller, and Hans Jürgen Holberg. 2016. Universal Pattern: formalization, testing, coverage, verification, and test case generation for safety-critical requirements. In *19. GI/ITG/GMM-Workshop "Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen" MBMV 2016*. https://doi.org/10.6094/UNIFR/10633

[57] Amit Kumar Tyagi and N. Sreenath. 2021. Cyber Physical Systems: Analyses, challenges and possible solutions. *Internet of Things and Cyber-Physical Systems* 1 (2021).

[58] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[59] Ben Wang. 2021. Mesh-Transformer-JAX: Model-Parallel Implementation of Transformer Language Model with JAX. https://github.com/kingoflolz/mesh-transformer-jax.

[60] Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax.

[61] Rongjie Yan, Chih-Hong Cheng, and Yesheng Chai. 2015. Formal consistency checking over specifications in natural languages. In *2015 Design, Automation Test in Europe Conference Exhibition (DATE)*.

[62] Aya Zaki-Ismail, Mohamed Osama, Mohamed Abdelrazek, John Grundy, and Amani Ibrahim. 2021. ARF: Automatic Requirements Formalisation Tool. In *2021 IEEE 29th International Requirements Engineering Conference (RE)*.

A Appendix

Table A.1. Support Set: 1 Example Logical Formulae (EQUAL).

Original	Translation
The duration of a flashing cycle is 1 second.	The duration of a flashing cycle is EQUAL 1 second.

Table A.2. Support Set: 1 Example Logical Formulae (LESS-OR-EQUAL).

Original	Translation
The maximum deviation of the pulse ratio should be below the cognitive threshold of a human observer	The maximum deviation of the pulse ratio should be LESS OR EQUAL cognitive threshold of human observer.

Table A.3. Support Set: 4 Examples Logical Formulae.

Original	Translation
The duration of a flashing cycle is 1 second.	The duration of a flashing cycle is EQUAL 1 second.
The maximum deviation of the pulse ratio should be below the cognitive threshold of a human observer.	The maximum deviation of the pulse ratio should be EQUAL cognitive threshold of human observer.
The minimal number of seatbelts used has to be 1.	The number of seatbelts used has to be GREATER OR EQUAL 1.
The vehicles doors are closed automaticly when speeding velocity is bigger than 10km/h.	The vehicles doors are closed automaticly when speeding velocity is GREATER 10km/h.

Table A.4. Support Set: 6 Examples Logical Formulae.

Original	Translation
The duration of a flashing cycle is 1 second.	The duration of a flashing cycle is EQUAL 1 second.
Also after 1000 flashing cycles the cumulated deviation must not exceed 0.05s.	After 1000 flashing cycles the cumulated Deviation must be GREATER OR EQUAL 0.05s.
The cruising speed has to be set at a speed which exceeds 10km/h.	The cruising speed has to be GREATER 10km/h.
The maximum deviation of the pulse ratio should be below the cognitive threshold of a human observer.	The maximum deviation of the pulse ratio should be EQUAL cognitive threshold of human observer.
The minimal number of seatbelts used has to be 1.	The number of seatbelts used has to be GREATER OR EQUAL 1.
The vehicles doors are closed automaticly when speeding velocity is bigger than 10km/h.	The vehicles doors are closed automaticly when speeding velocity is GREATER 10km/h.

Table A.5. Support Set: 8 Examples Logical Formulae.

Original	Translation
The duration of a flashing cycle is 1 second.	The duration of a flashing cycle is EQUAL 1 second.
Also after 1000 flashing cycles the cumulated deviation must not exceed 0.05s.	After 1000 flashing cycles the cumulated Deviation must be GREATER OR EQUAL 0.05s.
The cruising speed has to be set at a speed which exceeds 10km/h.	The cruising speed has to be GREATER 10km/h.
The maximum deviation of the pulse ratio should be below the cognitive threshold of a human observer.	The maximum deviation of the pulse ratio should be EQUAL cognitive threshold of human observer.
The minimal number of seatbelts used has to be 1.	The number of seatbelts used has to be GREATER OR EQUAL 1.
The vehicles doors are closed automaticly when speeding velocity is bigger than 10km/h.	The vehicles doors are closed automaticly when speeding velocity is GREATER 10km/h.
The interior material must be Velour.	The interior material is EQUAL Velour.
Flashing light cycle does not exceed 3 flashes.	Flashing light cycle is LESS or EQUAL 3 flashes.

Table A.6. Evaluation Results: 1 Example Logical Formulae (EQUAL).

Original	Translation	Class
The luminous intensity of the daytime running light must be lower than 400cd.	The luminous intensity of the daytime running light must be EQUAL 400cd.	6
The range of the vehicle shall be at least 450km under WLTP conditions	The range of the vehicle shall be at least 450km under WLTP conditions.	5
If the speed limiter is activated, the vehicle speed shall not exceed the set speed.	If the speed limiter is activated, the vehicle speed shall not exceed the set speed.	5
The maximum curb weight of the vehicle must be no more than 3.5t	The maximum curb weight of the vehicle must be EQUAL 3.5t.	3
The breaking distance can not be longer than 300m.	The breaking distance can not be longer than 300m.	5
The vehicles horn must not be louder than 50dz.	The vehicles horn must not be louder than 50dz.	5
The vehicles maximum velocity is 260km/h.	The vehicles maximum velocity is EQUAL 260km/h.	1
The minimun distance to a vehicle in front has to be 5m.	The minimun distance to a vehicle in front has to be EQUAL 5m.	1

Table A.7. Evaluation Results: 1 Example Logical Formulae (LESS-OR-EQUAL).

Original	Translation	Class
The luminous intensity of the daytime running light must be lower than 400cd.	The luminous intensity of the daytime running light must be LESS OR EQUAL 400cd.	3
The range of the vehicle shall be at least 450km under WLTP conditions	The range of the vehicle shall be at least 450km under WLTP conditions.	5
If the speed limiter is activated, the vehicle speed shall not exceed the set speed.	If the speed limiter is activated, the vehicle speed shall not exceed the set speed.	5
The maximum curb weight of the vehicle must be no more than 3.5t	The maximum curb weight of the vehicle must be LESS THAN 3.5t.	3
The breaking distance can not be longer than 300m.	The breaking distance can not be longer than 300m.	5
The vehicles horn must not be louder than 50dz.	The vehicles horn must not be louder than 50dz.	5
The vehicles maximum velocity is 260km/h.	The vehicles maximum velocity is LESS OR EQUAL 260km/h.	3
The minimun distance to a vehicle in front has to be 5m.	The minimum distance to a vehicle in front has to be LESS OR EQUAL 5m.	3

Table A.8. Evaluation Results: 4 Examples Logical Formulae.

Original	Translation	Class
The luminous intensity of the daytime running light must be lower than 400cd.	The luminous intensity of the daytime running light must be LESS THAN 400cd.	1
The range of the vehicle shall be at least 450km under WLTP conditions	The range of the vehicle shall be at least 450km under WLTP conditions.	5
If the speed limiter is activated, the vehicle speed shall not exceed the set speed.	If the speed limiter is activated, the vehicle speed shall not exceed the set speed.	5
The maximum curb weight of the vehicle must be no more than 3.5t	The maximum curb weight of the vehicle must be LESS THAN 3.5t.	3
The breaking distance can not be longer than 300m.	The breaking distance can not be LESS THAN 300m.	6
The vehicles horn must not be louder than 50dz.	The vehicles horn must not be LOUDER THAN 50dz.	5
The vehicles maximum velocity is 260km/h.	The vehicles maximum velocity is LESS THAN 260km/h.	6
The minimun distance to a vehicle in front has to be 5m.	The minimun distance to a vehicle in front has to be LESS THAN 5m.	6

Table A.9. Evaluation Results: 6 Examples Logical Formulae.

Original	Translation	Class
The luminous intensity of the daytime running light must be lower than 400cd.	The luminous intensity of the daytime running light must be LESS THAN 400cd.	1
The range of the vehicle shall be at least 450km under WLTP conditions	The range of the vehicle shall be at least 450km under WLTP conditions.	5
If the speed limiter is activated, the vehicle speed shall not exceed the set speed.	If the speed limiter is activated, the vehicle speed shall not exceed the set speed.	5
The maximum curb weight of the vehicle must be no more than 3.5t	The maximum curb weight of the vehicle must be LESS OR EQUAL 3.5t.	1.
The breaking distance can not be longer than 300m.	The breaking distance can not be LESS OR EQUAL 300m.	6
The vehicles horn must not be louder than 50dz.	The vehicles horn must be LESS OR EQUAL 50dz.	3
The vehicles maximum velocity is 260km/h.	The vehicles maximum velocity is LESS OR EQUAL 260km/h.	3
The minimun distance to a vehicle in front has to be 5m.	The minimum distance to a vehicle in front has to be LESS OR EQUAL 5m.	3

Table A.10. Evaluation Results: 8 Examples Logical Formulae.

Original	Translation	Class
The luminous intensity of the daytime running light must be lower than 400cd.	The luminous intensity of the daytime running light must be LESS THAN 400cd.	1
The range of the vehicle shall be at least 450km under WLTP conditions	The range of the vehicle shall be at least 450km under WLTP conditions.	5
If the speed limiter is activated, the vehicle speed shall not exceed the set speed.	If the speed limiter is activated, the vehicle speed shall not exceed the set speed.	5
The maximum curb weight of the vehicle must be no more than 3.5t	The maximum curb weight of the vehicle must be LESS THAN 3.5t.	1.
The breaking distance can not be longer than 300m.	The breaking distance can not be GREATER THAN 300m.	1
The vehicles horn must not be louder than 50dz.	The vehicles horn must not be LOUDER THAN 50dz.	5
The vehicles maximum velocity is 260km/h.	The vehicles maximum velocity is LESS THAN 260km/h.	6
The minimun distance to a vehicle in front has to be 5m.	The minimun distance to a vehicle in front has to be LESS THAN 5m.	6

Table A.11. Support Set: 1 Example If-Then Structure.

Original	Translation
If a defective illuminant is detected, the information about the defective illuminant is transmitted to the instrument cluster.	IF: defective illuminant is detected, THEN: information about the defective illuminant is transmitted to the instrument cluster

Table A.12. Support Set: 4 Examples If-Then Structure.

Original	Translation
If a defective illuminant is detected, the information about the defective illuminant is transmitted to the instrument cluster.	IF: defective illuminant is detected, THEN: information about the defective illuminant is transmitted to the instrument cluster
If no advancing vehicle is recognized any more, the high beam illumination is restored within 2 seconds.	IF: no advancing vehicle is recognized any more, THEN: high beam illumination is restored within 2 seconds.
If the light rotary switch is in position "auto", the adaptive high beam headlights are activated by moving the pitman arm to the back.	IF: the light rotary switch is in position ""auto"" and the pitman arm is moved back, THEN: the adaptive high beam headlights are activated.
(a) When the driver enables the cruise control (by pulling the cruise control lever or by pressing the cruise control lever up or down), the vehicle maintains the set speed if possible.	IF: driver enables the cruise control by pulling the cruise control lever or by pressing the cruise control lever up or down, THEN: the vehicle maintains the set speed if possible.

Table A.13. Support Set: 6 Examples If-Then Structure.

Original	Translation
If a defective illuminant is detected, the information about the defective illuminant is transmitted to the instrument cluster.	IF: defective illuminant is detected, THEN: information about the defective illuminant is transmitted to the instrument cluster
If no advancing vehicle is recognized any more, the high beam illumination is restored within 2 seconds.	IF: no advancing vehicle is recognized any more, THEN:high beam illumination is restored within 2 seconds.
If the light rotary switch is in position "auto", the adaptive high beam headlights are activated by moving the pitman arm to the back.	IF: the light rotary switch is in position ""auto"" and the pitman arm is moved back, THEN: the adaptive high beam headlights are activated.
(a) When the driver enables the cruise control (by pulling the cruise control lever or by pressing the cruise control lever up or down), the vehicle maintains the set speed if possible.	IF: driver enables the cruise control by pulling the cruise control lever or by pressing the cruise control lever up or down, THEN: the vehicle maintains the set speed if possible.
If the driver pushes down the cruise control lever with cruise control activated up to the first resistance level, the speed set point of the cruise control is reduced by N.	IF: driver pushes down the cruise control lever with cruise control activated up to the first resistance level, THEN: the speed set point of the cruise control is reduced by N.
By pushing the brake or the hand brake, the cruise control is deactivated until it is activated again.	IF: the brake or the hand brake is pushed, THEN: the cruise control is deactivated until it is activated again.

Table A.14. Evaluation Results: 1 Example If-Then Structure.

Original	Translation	Class
With activated darkness switch (only armored vehicles) the cornering light is not activated.	IF: darkness switch is activated, THEN: cornering light is not activated	3
When moving the pitman arm in position turn left the vehicle flashes all left direction indicators (front left, exterior mirror left, rear left) synchronically with pulse ratio bright to dark 1:1.	When moving the pitman arm in position turn left the vehicle flashes all left direction indicators (front left, exterior mirror left, rear left) synchronically with pulse ratio bright to dark 1:1.	5
If the driver activates the pitman arm during the three flashing cycles of tip-blinking again, only the current flashing cycle is completed.	IF: the driver activates the pitman arm during the three flashing cycles of tip-blinking, THEN: only the current flashing cycle is completed	1
Hazard Warning Light: As long as the hazard warning light switch is released, all direction indicators flash synchronically. If the ignition key is in the ignition lock, the pulse ratio is bright to dark 1:1. If the ignition key is not in the lock, the pulse ratio is 1:2.	IF: hazard warning light is released, THEN: all direction indicators flash synchronically. IF: ignition key is in the ignition lock, THEN: pulse ratio is bright to dark 1:1. IF: ignition key is not in the lock, THEN: pulse ratio is 1:2.	6
When hazard warning is deactivated and the pit arm is in position direction blinking left or direction blinking right, the direction blinking cycle should be released.	WHEN: hazard warning is deactivated, THEN: direction blinking cycle should be released	5
If tip-blinking was activated shortly before deactivation of the hazard warning, this is not considered during the deactivation of the hazard warning	IF: tip-blinking was activated shortly before deactivation of the hazard warning, THEN: this is not considered during the deactivation of the hazard warning	1
(a) The driver activates the low beam headlights by turning the light rotary switch to position exterior lights on	(a) The driver activates the low beam headlights by turning the light rotary switch to position exterior lights on	5
The driver activates the low beam headlights by turning the light rotary switch to position exterior lights on	The driver activates the low beam headlights by turning the light rotary switch to position exterior lights on	5
Distance Warning: The vehicle warns the driver visually and/or acoustically if the vehicle is closer to the car ahead than allowed by the safety distance.	IF: distance warning is activated, THEN: the vehicle warns the driver visually and/or acoustically	6
Emergency Brake Assist: The vehicle decelerates in critical situations to a full standstill	Emergency Brake Assist: The vehicle decelerates in critical situations to a full standstill	5
If the driver pushes the cruise control lever up to the first resistance level and the cruise control is activated, the speed set point of the cruise control is increased by the value N.	IF: the driver pushes the cruise control lever up to the first resistance level, THEN: the speed set point of the cruise control is increased by the value N	5

Table A.15. Evaluation Results: 4 Examples If-Then Structure.

Original	Translation	Class
With activated darkness switch (only armored vehicles) the cornering light is not activated.	IF: with activated darkness switch, THEN: the cornering light is not activated.	4
When moving the pitman arm in position turn left the vehicle flashes all left direction indicators (front left, exterior mirror left, rear left) synchronically with pulse ratio bright to dark 1:1.	IF: when moving the pitman arm in position turn left the vehicle flashes all left direction indicators (front left, exterior mirror left, rear left) synchronically with pulse ratio bright to dark 1:1, THEN: the vehicle flashes all left direction indicators (front left, exterior mirror left, rear left) synchronically with pulse ratio bright to dark 1:1.	5
If the driver activates the pitman arm during the three flashing cycles of tip-blinking again, only the current flashing cycle is completed.	IF: the driver activates the pitman arm during the three flashing cycles of tip-blinking, THEN: only the current flashing cycle is completed.	1
Hazard Warning Light: As long as the hazard warning light switch is released, all direction indicators flash synchronically. If the ignition key is in the ignition lock, the pulse ratio is bright to dark 1:1. If the ignition key is not in the lock, the pulse ratio is 1:2.	IF: hazard warning light switch is released, THEN: all direction indicators flash synchronically. If the ignition key is in the ignition lock, the pulse ratio is bright to dark 1:1. If the ignition key is not in the lock, the pulse ratio is 1:2.	1
When hazard warning is deactivated and the pit arm is in position direction blinking left or direction blinking right, the direction blinking cycle should be released.	IF: hazard warning is deactivated and the pit arm is in position direction blinking left or direction blinking right, THEN: the direction blinking cycle should be released.	1
If tip-blinking was activated shortly before deactivation of the hazard warning, this is not considered during the deactivation of the hazard warning	IF: tip-blinking was activated shortly before deactivation of the hazard warning, THEN: this is not considered during the deactivation of the hazard warning.	1
(a) The driver activates the low beam headlights by turning the light rotary switch to position exterior lights on	IF: the driver activates the low beam headlights by turning the light rotary switch to position exterior lights on, THEN: the low beam headlights are activated.	2
The driver activates the low beam headlights by turning the light rotary switch to position exterior lights on	IF: the driver activates the low beam headlights by turning the light rotary switch to position exterior lights on, THEN: the low beam headlights are activated.	2
Distance Warning: The vehicle warns the driver visually and/or acoustically if the vehicle is closer to the car ahead than allowed by the safety distance.	IF: distance warning is activated, THEN: the vehicle warns the driver visually and/or acoustically.	6
Emergency Brake Assist: The vehicle decelerates in critical situations to a full standstill	IF: emergency brake assist is activated, THEN: the vehicle decelerates in critical situations to a full standstill	6
If the driver pushes the cruise control lever up to the first resistance level and the cruise control is activated, the speed set point of the cruise control is increased by the value N.	IF: the driver pushes the cruise control lever up to the first resistance level and the cruise control is activated, THEN: the speed set point of the cruise control is increased by the value N.	1

Table A.16. Evaluation Results: 6 Examples If-Then Structure.

Original	Translation	Class
With activated darkness switch (only armored vehicles) the cornering light is not activated.	IF: with activated darkness switch (only armored vehicles), THEN: the cornering light is not activated.	2
When moving the pitman arm in position turn left the vehicle flashes all left direction indicators (front left, exterior mirror left, rear left) synchronically with pulse ratio bright to dark 1:1.	IF: the pitman arm is moved in position turn left, THEN: the vehicle flashes all left direction indicators (front left, exterior mirror left, rear left) synchronically with pulse ratio bright to dark 1:1.	1
If the driver activates the pitman arm during the three flashing cycles of tip-blinking again, only the current flashing cycle is completed.	IF: the driver activates the pitman arm during the three flashing cycles of tip-blinking, THEN: only the current flashing cycle is completed	1
Hazard Warning Light: As long as the hazard warning light switch is released, all direction indicators flash synchronically. If the ignition key is in the ignition lock, the pulse ratio is bright to dark 1:1. If the ignition key is not in the lock, the pulse ratio is 1:2.	IF: the hazard warning light switch is released, THEN: all direction indicators flash synchronically. If the ignition key is in the ignition lock, the pulse ratio is bright to dark 1:1. If the ignition key is not in the lock, the pulse ratio is 1:2.	1
When hazard warning is deactivated and the pit arm is in position direction blinking left or direction blinking right, the direction blinking cycle should be released.	IF: hazard warning is deactivated and the pit arm is in position direction blinking left or direction blinking right, THEN: the direction blinking cycle should be released.	1
If tip-blinking was activated shortly before deactivation of the hazard warning, this is not considered during the deactivation of the hazard warning	IF: tip-blinking was activated shortly before deactivation of the hazard warning, THEN: this is not considered during the deactivation of the hazard warning	1
(a) The driver activates the low beam headlights by turning the light rotary switch to position exterior lights on	IF: the driver activates the low beam headlights by turning the light rotary switch to position exterior lights on, THEN: the low beam headlights are activated.	2
The driver activates the low beam headlights by turning the light rotary switch to position exterior lights on	IF: the driver activates the low beam headlights by turning the light rotary switch to position exterior lights on, THEN: the low beam headlights are activated.	2
Distance Warning: The vehicle warns the driver visually and/or acoustically if the vehicle is closer to the car ahead than allowed by the safety distance.	IF: distance warning is activated, THEN: the vehicle warns the driver visually and/or acoustically.	6
Emergency Brake Assist: The vehicle decelerates in critical situations to a full standstill	IF: emergency brake assist is activated, THEN: the vehicle decelerates in critical situations to a full standstill.	6
If the driver pushes the cruise control lever up to the first resistance level and the cruise control is activated, the speed set point of the cruise control is increased by the value N.	IF: the driver pushes the cruise control lever up to the first resistance level and the cruise control is activated, THEN: the speed set point of the cruise control is increased by the value N.	1

Table A.17. Support Set: 1 Example Modal Verbs.

Original	Translation
The maximum deviation of the pulse ratio should be below the cognitive threshold of a human observer.	The maximum deviation of the pulse ratio MUST be below the cognitive threshold of a human observer.

Table A.18. Support Set: 4 Examples Modal Verbs.

Original	Translation
The maximum deviation of the pulse ratio should be below the cognitive threshold of a human observer.	The maximum deviation of the pulse ratio MUST be below the cognitive threshold of a human observer.
Direction blinking: For USA and CANADA, the daytime running light shall be dimmed by 50% during direction blinking on the blinking side.	Direction blinking: For USA and CANADA, the daytime running light MUST be dimmed by 50% during direction blinking on the blinking side.
The adaptation of the pulse ratio has to occur at the latest after two complete flashing cycles.	The adaptation of the pulse ratio MUST occur at the latest after two complete flashing cycles.
The duration of a flashing cycle is 1 second.	The duration of a flashing cycle MUST be 1 second.

Table A.19. Support Set: 6 Examples Modal Verbs.

Original	Translation
The maximum deviation of the pulse ratio should be below the cognitive threshold of a human observer.	The maximum deviation of the pulse ratio MUST be below the cognitive threshold of a human observer.
Direction blinking: For USA and CANADA, the daytime running light shall be dimmed by 50% during direction blinking on the blinking side.	Direction blinking: For USA and CANADA, the daytime running light MUST be dimmed by 50% during direction blinking on the blinking side.
The adaptation of the pulse ratio has to occur at the latest after two complete flashing cycles.	The adaptation of the pulse ratio MUST occur at the latest after two complete flashing cycles.
The duration of a flashing cycle is 1 second.	The duration of a flashing cycle MUST be 1 second.
A flashing cycle (bright to dark) will always be completed, before a new flashing cycle can occur.	A flashing cycle (bright to dark) MUST always be completed, before a new flashing cycle can occur.
With subvoltage the ambient light is not available.	With subvoltage the ambient light MUST not be available.

Table A.20. Evaluation Results: 1 Example Modal Verbs.

Original	Translation	Class
The reaction time between the activation of the pitman arm and the beginning of the first flashing cycle should be below the cognitive threshold of a human observer.	The reaction time between the activation of the pitman arm and the beginning of the first flashing cycle MUST be below the cognitive threshold of a human observer.	1
The frame rate of the camera is 60 Hz	The frame rate of the camera MUST be 60 Hz	1
The resolution of the camera signal is 24 bits (8 bits each RGB).	The resolution of the camera signal MUST be 24 bits (8 bits each RGB).	1
The functions of the system are classified as safety relevant in with respect to ISO 26262.	The functions of the system are classified as safety relevant in with respect to ISO 26262	5
Also after 1000 flashing cycles the cumulated deviation will not exceed 0.05s.	Also after 1000 flashing cycles the cumulated deviation will not exceed 0.05s.	5
Speed Limit: The vehicle does not exceed a set speed.	The vehicle does not exceed a set speed	5
A flashing cycle (bright to dark) has to be always completed, before a new flashing cycle can occur.	A flashing cycle (bright to dark) has to be always completed, before a new flashing cycle can occur.	5
Low beam illuminant shall be LED.	Low beam illuminant shall be LED.	5

Table A.21. Evaluation Results: 4 Examples Modal Verbs.

Original	Translation	Class
The reaction time between the activation of the pitman arm and the beginning of the first flashing cycle should be below the cognitive threshold of a human observer.	The reaction time between the activation of the pitman arm and the beginning of the first flashing cycle MUST be below the cognitive threshold of a human observer.	1
The frame rate of the camera is 60 Hz	The frame rate of the camera MUST be 60 Hz	1
The resolution of the camera signal is 24 bits (8 bits each RGB).	The resolution of the camera signal MUST be 24 bits (8 bits each RGB).	1
The functions of the system are classified as safety relevant in with respect to ISO 26262.	The functions of the system are classified as safety relevant in with respect to ISO 26262.	5
Also after 1000 flashing cycles the cumulated deviation will not exceed 0.05s.	Also after 1000 flashing cycles the cumulated deviation MUST NOT exceed 0.05s.	1
Speed Limit: The vehicle does not exceed a set speed.	The vehicle does not exceed a set speed	5
A flashing cycle (bright to dark) has to be always completed, before a new flashing cycle can occur.	A flashing cycle (bright to dark) MUST be always completed, before a new flashing cycle can occur.	1
Low beam illuminant shall be LED.	Low beam illuminant shall be LED.	5

Table A.22. Evaluation Results: 6 Examples Modal Verbs.

Original	Translation	Class
The reaction time between the activation of the pitman arm and the beginning of the first flashing cycle should be below the cognitive threshold of a human observer.	The reaction time between the activation of the pitman arm and the beginning of the first flashing cycle MUST be below the cognitive threshold of a human observer.	1
The frame rate of the camera is 60 Hz	The frame rate of the camera MUST be 60 Hz	1
The resolution of the camera signal is 24 bits (8 bits each RGB).	The resolution of the camera signal MUST be 24 bits (8 bits each RGB).	1
The functions of the system are classified as safety relevant in with respect to ISO 26262.	The functions of the system are classified as safety relevant in with respect to ISO 26262.	5
Also after 1000 flashing cycles the cumulated deviation will not exceed 0.05s.	Also after 1000 flashing cycles the cumulated deviation MUST not exceed 0.05s.	1
Speed Limit: The vehicle does not exceed a set speed.	The vehicle does not exceed a set speed.	5
A flashing cycle (bright to dark) has to be always completed, before a new flashing cycle can occur.	A flashing cycle (bright to dark) MUST always be completed, before a new flashing cycle can occur.	1
Low beam illuminant shall be LED.	Low beam illuminant MUST be LED.	1