R2BC : Tool-Based Requirements Preparation for Delta Analyses by Conversion into Boilerplates

Konstantin Zichler
Advanced Engineering Projects
HELLA GmbH & Co. KGaA
Lippstadt, Germany
konstantin.zichler@hella.com

Steffen Helke
Safe and Secure Software Systems
Brandenburg University of Technology
Cottbus-Senftenberg, Germany
steffen.helke@b-tu.de

***Abstract*—Automotive OEMs and suppliers negotiate different documents before they sign contracts for a product development. This includes the Component Requirements Specification (CRS), which is submitted by the OEM. The CRS describes the characteristics of the product to be developed in detail and is therefore the basis for the development effort estimation of a supplier. If the specified component is a successor of an already available product, the requirements specifications of both the successor and the predecessor products can be compared to estimate the development effort for the new component. This activity is called delta analysis. Due to a lack of sufficient tool support, the delta analysis is still a predominantly manual task. The main reason for this is, that the documents to be compared are structurally too different. In this work, we introduce a new method for an automated conversion of an OEM's unstructured or otherwise structured CRS into a structured language used by the supplier. The process uses established NLP tools to analyze CRS and then translates the OEM's requirements into supplier-specific boilerplates using a newly developed technique. The concept is implemented with the R2BC prototype, which demonstrates the feasibility of the approach and enables the processing of first real CRS.**

***Index Terms*—Requirements engineering, boilerplates, natural language processing, delta analysis**

I. Introduction

During the early phase of sourcing, OEMs submit Requests for Quotation (RFQ) to automotive suppliers. Among other documents, this request includes the Component Requirements Specifications (CRS). The CRS describes the properties of the component, which shall be developed. The RFQ prompts the supplier to offer the specified component at a certain price within a limited amount of time. In order to provide the OEM with an offer, the supplier has to first estimate the necessary effort to develop the requested component. If the supplier has already developed similar parts in the past, the former specification documents can be used to estimate the development effort. In this case requirements engineers perform a delta analysis. The delta analysis refers to the activity of comparing two requirements specifications to determine the differences, namely the deltas, between the listed requirements. It is a common procedure, in case a successor of an already available product is to be developed and the requirements specifications of both the successor and the predecessor products are available. The results of the comparison between both requirements specifications are used for the estimation of the effort necessary, to realize the successor product. The advantage of this approach is that the effort for the realization of the predecessor product is already known, and hence the effort for the adaptations, which are necessary to realize the successor product can be estimated. The below listed example illustrates a deviation of two similar requirements :

1) *If the combustion engine is running, the function ECU self-diagnosis shall be active.*
2) *If the vehicle battery is charging, the function ECU self-diagnosis shall be active.*

The underlined parts of the requirements sentences highlight the delta. In this example the additional working time of the ECU self-diagnosis function is to be considered. This can lead to additional development effort and may even require new components within the vehicle. These add-ons are subsequently subject to effort estimation activities of the supplier and hence the basis for price indication for the RFQ.

Due to a lack of sufficient tool support, the delta analysis is still a predominantly manual activity. Requirements Engineers experience the comparison of two documents with roughly 100 to 300 or more pages each, as tedious and time-consuming. In our opinion this time should rather be invested in creative work, which produces higher value-added for the company. It is for this reason that our development activities are focused on a novel approach for an automated delta analysis. From the experience we gained during our previous work [13], we know that an automated delta analysis would attain a higher accuracy, if sentences that are compared with each other have the same syntax.

A continuously equal syntax can be reached during the documentation of requirements by usage of boilerplates. A boilerplate is a blueprint that determines the syntactical structure of a single requirement [11]. Boilerplates compliant requirements have the same structure, if the same type of boilerplates is used. That means that certain elements of a sentence appear in the same order within all requirements of the same type. This fact enables a machine to compare certain parts of requirements and to determine the deviation, or in other words, the delta. In practice, companies use different boilerplates. Before requirements engineers of the supplier can benefit from an automated delta analysis, the submitted OEM requirements have to be converted into boilerplates compliant requirements first. This is especially the case, when requirements engineers at the supplier side use company-specific boilerplates. It is important to mention, that the time schedule for responding to an RFQ is very tight. This fact makes a manual conversion of requirements into boilerplates unfeasible.

A convenient solution for this problem can only be achieved by a suitable tool support. This tool support should be economically reasonable by requiring the least amount of time and personnel deployment for the conversion to boilerplates. Besides these requirements, further challenges for the tool support arise from the quality of the submitted requirements. In literature certain quality criteria are known : atomicity, correctness, completeness, unambiguousness etc. [8]. Requirements boilerplates by their mere structure are designed to support requirements quality [11]. Many random natural language requirements do not meet these quality criteria and hence do not fit accurately into the predefined boilerplates. Hence, natural language requirements have to be reshaped by the tool support, before they can be converted into boilerplates. Table I gives an overview of the tasks, that should be accomplished during the conversion of random natural language requirements into boilerplates. In the following, we elaborate on selected examples :

1) *Restructuring of a sentence :* Example 1 (Table I) shows an input requirement, which is written in the passive form. The conversion of this requirement into an active form requires a rearrangement of the sentence parts. The output requirement after the conversion is read as follows : "The ECU shall monitor the liquid temperature."
2) *Adaptation of words :* Once the sentence parts of a requirement were rearranged, some words of this requirement need adaptation (see Example 2 in Table I). The phrase : "The temperature of the liquid.", previously started with a capital letter. Now this phrase is located at the middle of the requirements sentence. This requires an adaptation of the word "The", which is now written with a lower-case letter "the".
3) *Atomization of requirements :* Example 3 (Table I) contains two requirements. This is proven by the two process words, which describe two different functions of the ECU. A proper conversion into boilerplates requires a split of the requirement into the following two requirements : (1) "The ECU shall monitor the liquid temperature." and (2) "The ECU shall store the liquid temperature data."
4) *Resolving co-reference :* The second sentence in Example 5 (Table I) : "It shall store the temperature data." addresses the ECU and by this constitutes a second requirement. The conversion of this second sentence into a requirement requires a specification of the corresponding actor. The accurate conversion of these to sentences leads to the following two requirements : (1) "The ECU shall monitor the liquid temperature." and (2) "The ECU shall store the liquid temperature data."

No.	Examples	
1	***Restructuring of a sentence (e.g. from passive to active form)***	
	Input	The liquid temperature shall be monitored by the ECU.
	Output	The ECU shall monitor the liquid temperature.
2	***Adaption of words***	
	Input	The temperature of the liquid shall be sent by the temperature sensor to the monitoring system.
	Output	The temperature sensor shall send the temperature of the liquid to the monitoring system.
3	***Atomization of requirements***	
	Input	The ECU shall monitor and store the liquid temperature.
	Output	The ECU shall monitor the liquid temperature. The ECU shall store the liquid temperature data.
4	***Identification on an actor in indefinite wording***	
	Input	It shall be ensured that the liquid temperature does not exceed the threshold of 120°C.
	Output	The ECU shall ensure that the liquid temperature does not exceed the threshold of 120°C.
5	***Resolving co-reference***	
	Input	The ECU shall monitor the liquid temperature. It shall store the temperature data.
	Output	The ECU shall monitor the liquid temperature. The ECU shall store the liquid temperature data.

TABLE I
EXAMPLES FOR CONVERSION TASKS

Among others, [6] and [7] present methods and tool support for the documentation of requirements with boilerplates right from the beginning. The majority of CRS in industry are based on different styles of boilerplates or do not comply to boilerplates at all. This means, that automotive suppliers receive already documented requirements with this kind of characteristics. Therefore, available approaches, which require the use of boilerplates during the documentation of requirements, cannot help to overcome issues that arise while handling finalized specifications.

It is for these reasons, that we suggest a semi-automated conversion of random natural language requirements to predefined boilerplates. Our tool the Requirements to Boilerplates Converter (R2BC) converts randomly formulated natural language requirements, which are provided in various document formats into predefined boilerplates. Alongside with the conversion, the R2BC concept aims at the rectification of requirements flaws to increases requirements quality. Our solution involves natural language processing (NLP) techniques and a proprietary developed prototype of the R2BC. Moreover, in this work we provide future users of the R2BC with the corresponding methodology for the realization of a semi-automated conversion of requirements into boilerplates. The basic aim of the technology presented in this work, is to make requirements machine-readable. Our tool, the R2BC is a prerequisite for the subsequent automated delta analysis. The automated delta analysis is part of our ongoing research activities. Within this work, we focus on the R2BC and its sub-ordinance into the broader methodology of an automated delta analysis.

The remainder of this work is structured as follows. Chapter II provides the reader with the fundamentals on boilerplates and NLP. We introduce the reader to the R2BC methodology in Chapter III. Results of preliminary experiments are presented and discussed in Chapter IV. In Chapter V, we summarized major publications on related work. Chapter VI summarizes our findings and gives an outlook on further research activities.

II. Fundamentals

A. Requirements Boilerplates

Boilerplates are used to improve requirements quality and to increase the degree of formalization of requirements. A boilerplate is a blueprint that determines the syntactical structure of a single requirement. This predefined sentence structure helps to prevent phrasing errors, like the passive form, while documenting requirements. Boilerplates can be handled easily by unexperienced authors to write accurate requirements [11]. The following example shows a company-specific boilerplate :

The complete system "<CompleteSystemName>"
shall ≪description≫.

This boilerplate can be used to document functional requirements. It consists of editable and non-editable parts. To document a functional requirement, the author replaces the *"<CompleteSystemName>"* part with the system name and the *≪description≫* part with the function description of the system under consideration. The phrase *"The complete system"* and the modal *"shall"* constitute the non-editable parts of the boilerplate. Company-specific boilerplates reflect specific needs coming from the requirements engineering processes applied by a certain company. For instance, the semiformal structure of boilerplate-compliant requirements allows to derive requirements models automatically and to further process these models for requirements verification. Alongside company-specific boilerplates, the well-known boilerplates introduced by Chris Rupp [11] and the EARS boilerplates [9] are used in the industry.

B. Natural Language Processing

In this work, we use NLP to analyze requirements text and to identify specific elements of the requirements sentence. NLP is a means to the computerized understanding, analysis, manipulation, and generation of natural language [9]. We use the General Architecture for Text Engineering (GATE) [3] to run natural language pre-processing on the requirements documents. GATE is an open-source software, which is mainly used to annotate text, either manually or automatically. A wide range of applications, which are called processing resources, are available under GATE. We explain the main processing resources for our application using the following example, given the following original requirement :

The ECU shall monitor the liquid temperature.

The tokenizer splits the text in tokens, like numbers, punctuation marks and words. "The", "ECU" or "." are tokens within the example sentence. In the following steps, these tokens can be used to analyze the text in more depth, e.g. with gazetteers. Gazetteers are used to recognize named entities in text. They consist of lists with numbers or names of entities, like cities, organizations or first names. Once a string in the text equals a string in a gazetteer, the named entity can be assigned. For this purpose, the string in the text receives an annotation called "Lookup". Another important application is the sentence splitter. This application splits the text in sentences. To this end, a gazetteer list with abbreviations is used to distinguish these from punctuation marks that mark the end of a sentence. Part-of-speech taggers determine the part-of-speech of a token and annotate it accordingly. Once annotation for all of this information are available, GATE can invoke JAPE (Java Annotation Patterns Engine) transducers. This tool searches for a predefined pattern in the text and then annotates this part of the text according to a predefined rule. For this search, the information regarding tokens, sentence splits etc. is used by the JAPE transducer [4].

In the given example, a pattern described by a JAPE rule could search for the sentence part between "shall" and the punctuation mark. The action rule would cause this part to be annotated for example with "description".

A second JAPE rule can be used subsequently to search for the "description" annotation. As a consequence, the underlying string "monitor the liquid temperature" of the "description" annotation can be transferred into the editable part of the boilerplate presented in Chapter II-B. Alongside with the complete system name "ECU", which would be recognized accordingly, the conversion would lead to the following result :

The complete system "ECU" shall monitor the liquid temperature.

During our previous work [13], we observed that JAPE rules work at a higher accuracy when applied to sentences that have an equal syntax. Experiences from practical work show nevertheless that requirements syntax varies a lot. NLP provides means to cope with these variances. The pre-processing of requirements in several steps allows to recognize relevant parts of requirements although presented in different syntaxes and to transfer these parts into boilerplates. Once all requirements are available in boilerplates further NLP can be performed with high accuracy, e.g. for an automated delta analysis.

III. Requirements to Boilerplates Converter

In this chapter we present our concept for the Requirements to Boilerplates Converter (R2BC). The R2BC is a prerequisite for an automated delta analysis. For this purpose, we first describe how the R2BC is integrated into the broader methodology of the automated delta analysis. The second part of this chapter describes the architecture of the R2BC. Within the third, we explain the natural language pre-processing component. The concept for the R2BC is presented in the fourth part of this chapter. We complete this chapter by describing the working methodology for the usage of the R2BC.

A. R2BC as Part of a Methodology for an Automated Delta Analysis

We suggest a novel approach for an automated delta analysis as depicted in Fig. 1. The process is triggered once an OEM submits a CRS to a supplier. During Step 1 of the process, the R2BC is used to convert the OEM natural language requirements to boilerplates, which are used by the supplier. Once the requirements of the OEM and the supplier have the same syntax our tool called Delta Analyzer (DA), performs the automated delta analysis (Step 2). As a result, the DA provides a report, which can be used to estimate the necessary effort for the realization of the successor product.

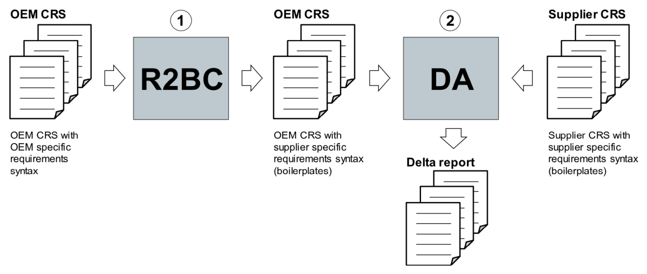

OEM CRS
1
R2BC
OEM CRS
2
DA
Supplier CRS
OEM CRS with OEM specific requirements syntax
OEM CRS with supplier specific requirements syntax (boilerplates)
Delta report
Supplier CRS with supplier specific requirements syntax (boilerplates)

FIGURE 1. Methodology for an automated delta analysis

B. R2BC Architecture

The architecture of the R2BC outlines three main components : a natural language pre-processing component, a converter and a GUI (see Fig. 2). The advantage of the R2BC is its flexibility. OEM CRS submitted to the supplier, are diverse in wording, caused by different authors or they differ in the document formats.

To cope with this fact, we implemented a natural language pre-processing component (NL pre-processing) into our tool. This component enables a flexible recognition of certain pieces of text and provides the input for the actual conversion into boilerplates. The centerpiece of the R2BC is called "Converter", our proprietary development. This component loads the input CRS from the file system of the computer and exports the CRS with the converted boilerplates to the file system. To this end, the Converter invokes the functionality of the NL pre-processing to recognize the necessary parts of the requirements text and then converts them into boilerplates. The Converter itself, is controlled by the GUI, used by the user to perform the

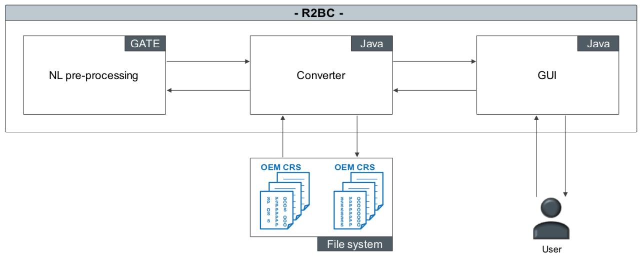


FIGURE 2. R2BC Implementation

required operations. All mentioned components of the R2BC are described in the following section, starting with NL pre-processing.

C. Natural Language Pre-processing

For the NLP we compiled a processing pipeline in GATE, which consists mainly of ANNIE [2] resources and several JAPE transducers. Fig. 3 gives an overview of the applied resources and the process.

Once the Converter invokes the NL pre-processing component, the following steps are performed :

1) The Converter loads the CRS of the OEM into GATE. This document is converted into a corpus, which is the basis for the NLP.
2) The corpus is analyzed by the components *Document Reset PR*, *English Tokeniser*, *Gazetteer*, *Sentence Splitter*, *POS Tagger*, *NE Transducer* and *OrthoMatcher* of the application ANNIE. We customized the mentioned components to our specific needs. ANNIE annotates the text and provides an annotated corpus as a result. These annotations provide mainly language-specific information.
3) During the third step, several *JAPE Transducers* uses *JAPE* rules to search annotations in the corpus. These *JAPE* rules are defined in advance and are intended to search for specific parts of the requirements sentences, which are transferred into the editable parts of boilerplates. First, *JAPE* rules determine, which requirement fits which boilerplate. Then, certain pieces of the annotated requirements are annotated according to the editable parts of boilerplates. All requirements and other statements in the CRS, which do not fit into boilerplates, receive a corresponding annotation. These sentences will be transferred into the export document without conversion into boilerplates.

The result of the NL pre-processing component is an annotated corpus, which is used by the Converter to convert requirements into boilerplates.

D. Converter

The Converter is the heart of the requirements to boilerplates conversion. This component loads documents into the NL pre-processing component and invokes the natural language

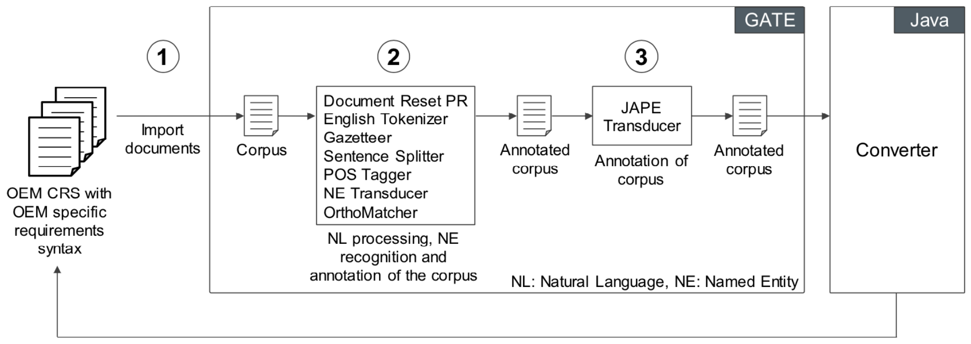


FIGURE 3. NL Pre-processing

analysis of the text. Once the NL pre-processing is finished, the Converter gathers the annotated text and searches among the contained annotations for text pieces which are to be converted to boilerplates. During the next step all requirements, which fit the applied boilerplates, are converted. The following two requirements give a simplified example for a conversion :

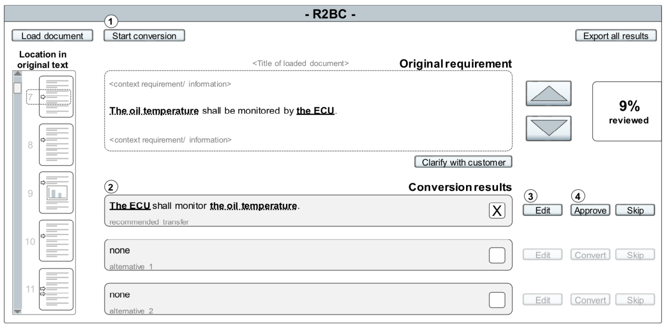
1) *The oil temperature shall be monitored by the ECU*
2) *The "ECU" shall monitor the oil temperature*

Requirement (1) is the original requirement. In course of the conversion, components of this requirement are rearranged in order and certain words are adapted automatically. Also, at this stage, the user can make adaptation to the conversion results. Due to the fact that some requirements can be converted to several boilerplates, several conversion results for these requirements are available. To this end, all conversion results are stored by the Converter for the moment. It is the user who ultimately decides which conversion alternative is correct. In our working methodology for the R2BC this step is called "Approve results". All steps of the working methodology are described in section III-E. After the user has approved all results of the conversion the results can be exported. The Converter exports the boilerplates compliant requirements to a format of choice. Among others, we consider the formats Word, PDF and the Requirements Interchange Format (ReqIF) [5] to enhance the work with requirements management tools.

E. R2BC Methodology

The R2BC allows a semi-automated conversion of natural language requirements into predefined boilerplates. The user interaction with the R2BC follows a four steps methodology : convert requirements, review results, edit results and approve results. We describe this methodology by means of the R2BC GUI depicted in Fig. 4.

Convert requirements: The user starts the conversion process for a CRS, which is already loaded into the application by pushing the "Start conversion" button (Fig. 4 Step 1). This triggers an algorithm, which invokes NLP to annotate the input CRS. The R2BC browses the created annotations and selects certain parts of the requirements text in the input CRS. As a consequence, all relevant requirements are converted automatically into boilerplates by the R2BC.



The oil temperature shall be monitored by the ECU.

The ECU shall monitor the oil temperature.

none

none

FIGURE 4. R2BC GUI

Review results: During the next step (Fig. 4 Step 2) the user reviews all converted requirements. For each converted requirement, the R2BC provides the user with the view of the original requirement and a view of alternatives for the conversion, called conversion results. In the original requirement view, the converted requirement is displayed in its original shape. In some cases, it is possible that one requirement fits several boilerplates. For this purpose, the R2BC shows all possible conversion alternatives. Hence, the user may select the most convenient alternative. Conversion results, which are incorrect can be skipped. The R2BC facilitates the review process by highlighting parts of the requirements, which were changed by the conversion. Changes involve syntactical structure of the requirements sentence as well as for upper and lower case and changes in word endings. By displaying the adjacent context of the original requirement in the original requirement view, the R2BC helps the user to evaluate the accuracy of the conversion result and to select the most convenient alternative.

Edit results: Within the edit results step (Figure 4 Step 3), the user can make adaptations to the conversion results. This may be necessary, if a conversion result has a defect. For this purpose, the user presses the edit button next to the corresponding conversion alternative. This activates the editing function within the conversion results view.

Approve results: Finally, the user approves all valid alternatives (Fig. 4 Step 4), whereby only one alternative per input requirement can be approved.

IV. Experiments and Discussion

This chapter presents the results of preliminary tests and feedback from requirements experts. It is based on the R2BC prototype, which was implemented and tested in an ongoing research project by Ritter und Schul [10].

A. Preliminary Experiments

The aim of the preliminary experiments was to evaluate the effectiveness of the R2BC prototype. The fully automated conversion is considered effective, when relevant requirements in a CRS are identified and converted into boilerplates accurately. A requirement is relevant for the conversion, if a corresponding boilerplate for this type of requirement is applied, e.g. a boilerplate for conditional requirements. We assessed the effectiveness of our prototype by calculating precision for the identification of relevant requirements in a given CRS. For all identified requirements, we also assessed the conversion accuracy. For this purpose, we calculated the percentage of the number of accurately converted requirements of all converted requirements.

Since in practice, different styles of CRS are submitted by OEMs, we have tested our prototype on three CRS from different requirements authors to increase the significance of the experimental findings. CRS 1 comprises 88 pages. CRS 2 is an extensive document with 567 pages and CRS 3 comprises 60 pages. All three documents consist mainly of requirements documented as complete and fragmented sentences.

As already mentioned, many companies define their own specific boilerplates. For our experiments, we have chosen the following two company specific boilerplates A and B :

A : The complete system "<CompleteSystemName>" shall ≪description≫.

B : [ELSE] IF <Condition>, THEN : [the function "<FunctionName>" shall [not]] ≪description≫ [ELSE : ≪description≫].

Boilerplate A is used to document functional requirements. B is a boilerplate for the documentation of conditional requirements. Hence, these two boilerplates determined, which requirements were relevant for the automated conversion during our experiments. It is important to note, that we did not distinguish between functional and non-functional requirements in our experiments. This means, if an identified non-functional requirement, was converted into boilerplate A, we considered it accurate, if it syntactically fit the boilerplate.

We used CRS 1 to manually analyze the sentence structure of the present requirements. Afterwards, these findings were used to define JAPE rules and to determine the settings of other NLP tools, which we implemented in the NL pre-processing component of the R2BC. Except for gazetteers, we applied the same tool settings for all three CRS. We applied CRS specific gazetteers for the conversion of requirements into boilerplate A. The gazetteers were mainly used to identify the system name, i.e. the subject in the original requirement. The conversion of requirements into boilerplate B was performed without gazetteer support. Within preliminary experiments the R2BC achieved sound results as presented in Table II.

The conversion of the 88 pages of CRS 1 into boilerplates A and B took roughly 10 seconds. The R2BC achieved a precision of 100% in identifying requirements, which are relevant for one of these two boilerplates. The conversion of identified requirements into boilerplate A worked with an accuracy of 62.5%. In the other 37.5% of conversion results, the system name was not converted completely into the boilerplate. In most of these cases the subject of the original requirement consisted of several words. The R2BC converted only part of the subject into the boilerplate. This was caused by the applied gazetteer, which contained several system names, of which some contained the same words. If for example, the gazetteer contained among others the system names "controller" and "controller module" and "controller module" was mentioned in the original requirement, the gazetteer recognized "controller" as system name and cut off "module".

Within the same CRS, the R2BC achieved a conversion accuracy of 89.3% for boilerplate B. The conversion accuracy was lowered by the diversity of wording, which was applied to express conditions. For instance, some authors used "In the event" instead of "If" or "When". Also spelling errors prevented a higher score. For instance, authors did not place a comma after the condition description in an if clause. We have also observed that some requirements contained several conditions. As a reminder, the R2BC did not use a gazetteer for the conversion of boilerplate B and still generated better results, than for boilerplate A. This leads us to the conclusion that the application of a gazetteer can also lead to a disadvantage. In addition to that, we calculated the recall score for CRS 1. The R2BC identified relevant requirements for boilerplate A with a recall of 100%. The recall score for boilerplate B with 66.7% was lowered by the same reasons as mentioned before.

We applied the same tool setup, except for the gazetteer, which we adapted accordingly, to the automated conversion of CRS 2. The conversion of this extensive document comprising 567 pages was accomplished within 89 seconds. Also, for this CRS the R2BC achieved a precision of 100% in identifying requirements, which are relevant for boilerplate A and boilerplate B. The conversion into boilerplate A worked with an accuracy of 93.3%. For boilerplate B, the R2BC achieved a conversion accuracy of 67.6%. These calculations are based on results for the first 400 pages of CRS 2. Since CRS 2 is an extensive document, we took the first 400 pages as a large sample and refrained from the rest. Only 6.7% of the conversion results for CRS 2 had defects. These defects were caused again by the partly recognition of the subject of the original requirement, which consisted of several words. In contrast, the R2BC achieved a conversion accuracy of 67.6% for boilerplate B. As for CRS 1, the main reasons for conversion flaws are multiple conditions per requirement and missing commas. The R2BC converted the 60 pages of CRS 3 within 7 seconds into boilerplates. As a result, the identification of requirements for the conversion into

	Results					
	Boilerplate A		***Boilerplate B***		***Further Information***	
	Precision of requirements identification	***Conversion accuracy***	***Precision of requirements identification***	***Conversion accuracy***	***Conversion time***	***Pages***
CRS 1	100%	62.5%	100%	89.3%	$10s$	88
CRS 2	100%	93.3%[1]	100%	67.6%[1]	$89s$	567
CRS 3	100%	100%	100%	84.6%	$7s$	60

[1]The calculations are based on results for the first 400 pages of CRS 2

TABLE II
RESULTS OF PRELIMINARY EXPERIMENTS

boilerplate A and B worked for both with a precision of 100%. The conversion of identified requirements into boilerplate A worked with an accuracy of 100%. For boilerplate B the R2BC achieved a conversion accuracy of 84.6%. The reason for conversion defects were several conditions per requirement.

In summary, the R2BC prototype automatically analyzes large amounts of requirements text and recognizes relevant requirements for the conversion to predefined boilerplates. Subsequently all relevant requirements are converted into boilerplates automatically by the R2BC. Preliminary experiments show sound results. Especially within the large CRS with 567 pages the R2BC achieved a conversion precision of 93.3%. This score makes the presented technology promising. Nevertheless, the recognition of sentence parts of the original requirements should be improved. We propose to adapt the JAPE rules to cope with multi word subjects. Although our JAPE rules already target system names consisting of several words, evidence shows that the number of tokens to be taken into account for a system name should be increased. We also plan to elaborate the combination of JAPE rules and gazetteer lists for a proper named entity recognition.

The results, which were presented so far were attained by the fully automated conversion. However the R2BC methodology is designed as a semi-automatic process, i.e. the user is able to check the results. The conversion results, which we considered incorrect in the above evaluation, in most cases just require minor adjustments. We presented our methodology to industry experts. Their feedback and their suggestions for improvement are described in the following section.

B. Validation with Industry Experts

We presented our methodology for the R2BC and our proprietary prototype to requirements engineers. Besides the automated conversion functionality, we implemented a selection of the functionality depicted in Fig 4 in our prototype. The prototype contains a screen for the original requirement and three other screens for the conversion alternatives. A user can interact with the R2BC prototype by using the "Load document", "Start conversion", "Confirm", "Skip" and "Export results" buttons. Also, it is possible to edit the suggested conversion alternatives manually. This tool concept was assessed by the expert group to be very useful.

Also, we presented the future GUI of the R2BC to industry experts, as illustrated in Fig 4. The general setup of this GUI was confirmed by the experts. Among others, experts recommended to implement a "Clarify with customer" button into the GUI. This button shall allow to store an unclear requirement in a separate list. This list of unclear requirements can be discussed with the OEM after the conversion. To make sure that the actual question regarding this kind of requirements will not get lost, the experts suggested to add a dialog box for taking notes, which should appear once the "Clarify with customer" button is clicked. This function shall allow to specify the unclear aspect of the requirement or to document a question. In case many notes were taken, this would allow the requirements engineer to remember the questions, when talking with the customer.

For further improvement, industry experts suggested to implement a functionality that allows the engineer to focus only on those conversion results that are likely to be defective. As shown by the conversion accuracy scores gained during preliminary experiments, most of the conversion results are correct and therefore do not need further adjustment. According to the suggestion of the experts an algorithm, that works in the background could calculate the probability of the correctness of the conversion results, which would last in a reliability measure. Hence, all conversion results above a certain threshold would be considered reliable and therefore would not need to be reviewed. Instead, only those conversion result that have a value below this threshold, should be reviewed by the requirements engineer. This function would allow to work more efficient with the R2BC.

In conclusion, industry experts assessed our proprietary developed R2BC prototype as useful and promising. Their feedback showed, that usability and efficiency are key for a successful implementation of the R2BC. Our next version of the R2BC will implement the presented suggestions for improvement alongside with other features, to serve practitioners best at their daily tasks.

V. RELATED WORK

Arora et al. developed an approach for the automated checking of conformance of natural language requirements to boilerplates based on NLP techniques. They introduce a generalizable method for casting templates into NLP pattern matchers. For this purpose, they translate common templates into a BNF (Backus-Naur form) grammar. Afterwards, these grammars are implemented as JAPE pattern matching rules for checking template conformance. According to Arora et al. the approach provides a robust and accurate basis for checking conformance to templates [1].

Farfelder et al. provide requirements engineers with predefined boilerplates and a domain ontology for the documentation of high-quality requirements during elicitation. To start the documentation with DODT, the requirements engineer uses the GUI and chooses from a set of predefined boilerplates. Subsequently DODT is accessing a domain ontology, which contains all available words for the editable parts of the chosen boilerplate. The requirements engineer selects the required words from the list and defines by this the requirements. DODT is based on NLP techniques [6].

Schraps and Bosler present an approach to extract knowledge from software requirements and to transfer it into a requirements ontology. They use NLP techniques to annotate requirements first. Second, a pattern recognition algorithm is searching for predefined patterns within the grammar of the requirements. As a consequence, all parts of the requirements which fit into these patterns are transferred into the requirements ontology. By this approach Schraps and Bosler are aiming at the elimination of inconsistencies between specification and software models [12].

Fockel et al. describe a methodology for the documentation of functional requirements with boilerplates. According to this methodology an overall function is decomposed into its leaf functions. The deployment of boilerplates together with this methodology leads to a complete model of the requirements specification. To enable an efficient deployment of the boilerplates and the methodology Fockel et al. developed a tool support, called ReqPat. ReqPat can be integrated in commercial tools like IBM Rational DOORS. This tool does not only support the user during the documentation of requirements, it also tests the quality of the requirements automatically. Moreover, ReqPat is able to transfer boilerplates compliant functional requirements into modeling tools (e.g. SysML/UML tools) [7].

None of the presented approaches enables a semi-automated conversion of random natural language requirements into predefined boilerplates. While [6] and [7] present methods and tool support for the documentation of requirements with boilerplates right from the beginning, our experience shows, that automotive suppliers receive requirements, which comply to different styles of boilerplates or do not comply to boilerplates at all. Natural language requirements have to be converted into boilerplates first, before one can benefit from their semiformal nature. The R2BC provides a flexible and efficient way to convert random requirements into predefined boilerplates. This is a preparatory stage for machine-readability. As a consequence, these requirements can be processed automatically in further product development processes, e.g. in an automated delta analysis. Moreover, the R2BC methodology aims at high usability. This will allow requirements engineers, who have no experience in NLP, to take advantage of this beneficiary technology.

VI. Summary and Outlook

In this work, we presented the Requirements to Boilerplates Converter (R2BC), which is a prerequisite for an automated delta analysis. The R2BC is a novel approach for a semi-automated conversion of random natural language requirements into predefined boilerplates. To achieve this task, we applied NLP and a proprietary developed converter. Alongside the technology, we provided future users with a methodology. During preliminary experiments the R2BC prototype processed large documents with up to 567 pages within seconds and achieved high precision in requirements identification and conversion accuracy scores. The sound results prove the effectiveness of our approach. In addition to that, industry experts evaluated our proprietary developed R2BC prototype and the methodology as highly useful and promising. Our future activities are focused on the improvement of the R2BC. To this end, we will use the conclusions from preliminary experiments and the feedback from industry experts. Above all, we will focus our effort on the development of a concept and tool support for an automated delta analysis.

References

[1] C. Arora and M. Sabetzadeh and L. Briand and F. Zimmer. Automated Checking of Conformance to Requirements Templates using Natural Language Processing. IEEE, 2015.

[2] H. Cunningham and D. Maynard and K. Bontcheva and V. Tablan. GATE : A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 2002)*, 2002.

[3] H. Cunningham and V. Tablan and A. Roberts and K. Bontcheva. Getting More Out of Biomedical Documents with GATE's Full Lifecycle Open Source Text Analytics. *PLOS Computational Biology*, **9(2)**, 2013.

[4] H. Cunningham and et al. Developing Language Processing Components with GATE Version 8 (User Guide). University of Sheffield, Department of CS, 2014.

[5] C. Ebert and M. Jastram. ReqIF : Seamless Requirements Interchange Format between Business Partners IEEE-Software, **29(5)**, 2012.

[6] S. Farfeleder and T. Moser and A. Krall and T. Stalhane and I. Omoronyia and H. Zojer. Ontology-Driven Guidance for Requirements Elicitation. Springer, LNCS 6644, 2011.

[7] M. Fockel and J. Holtmann and M. Meyer. Mit Satzmustern hochwertige Anforderungsdokumente effizient erstellen. In OBJEKTspektrum, RE/2014, 2014.

[8] ISO, IEC, and IEEE. ISO/IEC/IEEE 29148. Technical report, ISO IEEE IEC, 2011.

[9] C. Manning and H. Schütze. Foundations of statistical natural language processing. MIT press, 1999.

[10] F. Ritter and A. Schul. Entwurf und Implementierung einer Werkzeugunterstützung zur sprachlichen Analyse und automatisierten Transformation von Projektlastenheften im Kontext der Automobilindustrie. Bachelor thesis, FH Dortmund, 2019.

[11] C. Rupp and SOPHIST-Gesellschaft für Innovatives Software-Engineering (Nürnberg). Requirements-Engineering und -Management, Aus der Praxis von klassisch bis agil. Hanser, 2014.

[12] M. Schraps and A. Bosler. Knowledge Extraction from German Automotive Software Requirements using NLP-Techniques and a Grammar-based Pattern Detection. In Proc. of the Int. Conf. on Pervasive Patterns and Applications, 2016.

[13] K. Zichler and S. Helke. Ontologiebasierte Abhängigkeitsanalyse im Projektlastenheft. In *Proceedings Automotive - Safety und Security (AUTOMOTIVE 2017)*, GI-LNI, **269**, 2017.