Should We Adopt a New Version of a Standard? – A Method and Its Evaluation on AUTOSAR

Corrado Motta[1](✉), Darko Durisic[1](✉), and Miroslaw Staron[2]

[1] Volvo Car Group, Gothenburg, Sweden
{Corrado.Motta,Darko.Durisic}@volvocars.com
[2] Chalmers University of Gothenburg, Gothenburg, Sweden
Miroslaw.Staron@cse.gu.se

Abstract. The development of large software systems is usually based on a number of industrial standards that define a set of features and their requirements. In order to use new features specified in the standards, new releases of the standards need to be adopted together with their requirements. This requires a thorough impact analysis of the changes in the requirements that can be time-consuming considering their potentially high number. In order to facilitate the adoption of new releases of industrial standards in large software systems, we present a method based on both quantitative and qualitative analysis of requirements evolution. The method is evaluated in a case study of AUTOSAR - a standard used in the development of automotive software systems in cooperation with Volvo Car Group. The evaluation results show that the use of the proposed method can identify the most unstable AUTOSAR specifications and their requirements whose changes may have a significant impact on the automotive systems. This knowledge can increase the speed of adoption of new AUTOSAR releases by automotive vendors.

Keywords: Requirement evolution · Metrics · Industrial standards

1 Introduction

Analyzing the evolution of system requirements is an important and inevitable phase in the development of large software systems [8], especially for OEM (Original Equipment Manufacturers) that base their development on industrial standards. This is because features specified in the standards and their requirements are usually driven by a number of competitive companies. The process of updating one system with new standardized features brings a series of advantages, such as making use of a number of standardized requirements that are proved to be valid in practice and buying cheaper off-the-shelf software packages from software vendors. However, it also brings new challenges such as working with requirements not written by OEMs and dealing with their fast evolution.

For this reason, the process of analyzing the evolution of standardized requirements without a suitable methodology and tool support can be time-consuming and can require significant engineering effort. Furthermore, adopting new standardized features in the development projects without having a clear knowledge whether the standardized requirements related to them can be fulfilled in practice can lead to the introduction of new faults into the system. Therefore, dealing with the evolution of standardized requirements is one of the primary objectives of large companies in order to be able to update their systems faster and cheaper.

Although several solutions have been proposed mostly originating from the academia, the evolution of standardized requirements is still considered one of the most challenging practical problems in the development of large software systems [8]. The objective of this work is to define a suitable methodology for efficiently analyzing the evolution of standardized requirements as well as improving the process of updating large software system with new standardized features. We aim to provide an answer to the following research question: *How can we assure efficient adoption of new releases of standards in the development of large software systems by analyzing the evolution of standardized requirements?*

As outcome of our work, this paper presents a method, named SREA (Standardized Requirements Evolution Assessment), that consists of four steps that are based on both quantitative and qualitative analysis of requirements evolution. All steps of the method can and should be performed automatically with the help of a software tool in order to reduce the time of analysis. Despite the fact that we focus on the analysis of standardized requirements where the proposed method is particularly helpful as certain requirements changes are driven by other companies, the method can also be applied on the evolution of company internal requirements as part of the common requirements engineering process.

In order to evaluate the proposed method, we applied it in the automotive domain in a case study of AUTOSAR (Automotive Open System Architecture) standard [3], that specifies a reference architecture and methodology for the development of automotive software systems. In particular, AUTOSAR provides a set of standardized requirements for the design of the automotive architectures (i.e., language for the architectural models), that consists of a number of Electronic Control Units (ECUs) responsible for executing software functionalities, and requirements for the ECU middleware. The rest of the functional ECU requirements are left to be defined by each OEM. As AUTOSAR represents a big industrial standard that counts more than 150 partners and 21.000 requirements, we believe it qualifies as a valid case for evaluating the *SREA* method.

The study is conducted in collaboration with Volvo Car Group (VCG) whose engineers helped us to understand and prioritize AUTOSAR releases and their specifications for the evaluation of the proposed method. The results of our evaluation show the importance of performing automated analysis of requirements evolution. In particular, they show that the use of *SREA* method could help automotive engineers in analyzing the evolution of the AUTOSAR requirements faster by providing the engineers with information such as which specifications are unstable (e.g., their requirements change with every release), which requirements are changed and the actual content of the changed requirements.

Knowledge about the unstable specification can be useful for making strategic decisions about the set of standardized features that are mature enough to be implemented in the system. Knowledge about the changed requirements has a potential of saving lots of time spent on reading tents of AUTOSAR specification with thousands of requirements manually. For example, in all specification with design requirements between AUTOSAR releases *4.2.1* and *4.2.2*, we identified 1563 requirements and presented relevant changes (additions, removals and modifications) for only 172.

The rest of the paper is organized as follows: Sect. 2 presents the related work; Sect. 3 provides the background of AUTOSAR as our unit of analysis; Sect. 4 describes the research methodology we used during this project; Sect. 5 defines the proposed *SREA* method; Sect. 6 presents the results of evaluation of the *SREA* method on AUTOSAR; Sect. 7 discusses the validation of the proposed method and provides recommendations; finally, Sect. 8 concludes our work and describes our plans for future work.

2 Related Work

Several studies of requirements evolution are related to our study. In particular, Wang et al. [15] provide a general method for studying changes in the requirements in order to find relations between them and the number of software defects. The method relies on the quantitative analysis. Although their approach was considered as a starting point for our work, our focus was on the impact of requirements changes on the system under development.

Similar but more exhaustive studies were conducted in the avionics context by Anderson et al. [1,2] who show how to conduct an empirical analysis of requirements evolution starting from a general point and moving to a product-oriented one. They perform the following two steps: (i) collecting information from the avionics domain using the *Requirement Maturity Index* (*RMI*), *Historical Requirement Maturity Index* (*HRMI*), and *Requirement Stability Index* (*RSI*) metrics and (ii) refining the information gathered in the first step using the qualitative approach. We rely on a similar work-flow based on the RMI metric that shows the stability of requirements changes in relation to the past releases. We did not consider RSI and HRMI metrics since a great number of requirements changes can lead to a non-meaningful (e.g., negative) number.

For the quantitative analysis of requirements evolution, we considered the study of Shi et al. [13] that aims to identify requirements that are most likely to be changed in the future using a number of metrics, e.g., *Sequence*, *Frequency* and *Lifecycle*. However, these metrics cannot be used for identifying the requirement specifications that are mostly affected by changes, that is one of our major goals.

For efficiently studying the evolution of system requirements, Nurmuliani et al. [11] provided a taxonomy of changes for categorizing different types of requirements changes, reasons for their change and the origin of changes. This study inspired us to define a taxonomy of changes for the AUTOSAR requirements.

Additionally, Stark g. et al. [14] proposed a method for analyzing the evolution of requirements in two steps. We adopted the first step called *micro* analysis in order to get a preliminary view on the requirements architecture, structure of the requirements specifications and possible types of requirements changes.

Finally, in order to understand the development of automotive software systems based on AUTOSAR, several automotive papers were considered. Two studies of Durisic et al. [6,7] were useful for improving our general knowledge of AUTOSAR, its architecture, methodology and complexity. More generally, the paper from Broy et al. [4] was useful for identifying trends in the evolution of the automotive software, even though they do not explain how it can be measured.

Although there is a significant number of methods related to the analysis of requirement evolution, the majority of them are not applied in a real industrial context in which large software systems are developed. This paper aims to fill this gap by combining the existing studies with our considerations and metrics in order to develop an efficient method for analyzing the evolution of standardized requirements that can facilitate this activity in the real industrial contexts.

3 Case Study Evaluation Context

In the automotive domain, OEMs are usually responsible only for the design of automotive systems while the actual implementation of software and hardware components is done by a hierarchy of suppliers. In order to standardize the methodology of work in such a distributed environment and a reference architecture for the distributed realization of the automotive systems using a number of ECUs, AUTOSAR standard has been introduced. The proposed ECU architecture is based on the following common three-layer architecture:

1. Application software layer that consists of a number of software components responsible for certain vehicle functionalities.
2. ECU middleware layer (a.k.a. basic software - BSW) that consists of a number of BSW modules responsible for, e.g., signaling and diagnostic services.
3. ECU hardware layer responsible for executing the allocated software components and BSW modules.

Since ECU basic software does not contribute to the realization of high level car functionalities that could make a competitive advantage for one OEM (e.g., autonomous drive), it is completely standardized by AUTOSAR that provides a set of requirements specifications for each BSW module. On the other hand, the functionality of the application layer is generally not standardized, although there are some predefined software components. However, AUTOSAR provides a meta-model followed by a set of design requirements that define the language for the architectural models of the application layer in order to facilitate the exchange of models between OEMs and suppliers [6]. This meta-model and design requirements serve as basis for the development of AUTOSAR modeling tools.

Based on the described methodology, we can distinguish between the following three types of requirements in the AUTOSAR based development process:

1. *Functional requirements* for the application software specified by OEMs.
2. *Design requirements* for the system models standardized by AUTOSAR.
3. *BSW requirements* for each BSW modules standardized by AUTOSAR.

In this paper, we focus on the last two types that are standardized by AUTOSAR. Design requirements are described in the specifications called "templates" (*TPS*) and they can be either *specification items* or *constraints* (checked by modeling tools). BSW requirements are described in the software requirement specifications *SWS*. An example for each type is provided below:

– **Specification item example:** The 1:n multicast routing is supported with the definition of several *IPduMappings* classes.
– **Constraint example:** The value of *windowSize* shall be greater or equal 1.
– **BSW requirement example:** CAN module shall allow that Multiplexed Transmission functionality is configurable (ON — OFF) at pre-compile time.

Apart from these types of requirements, AUTOSAR builds a requirements traceability hierarchy starting from the explained requirements until the general AUTOSAR features and objectives. These requirements are, together with the OEM-specific functional requirements, not considered in our analysis as they do not produce a direct impact on the development of automotive systems.

All AUTOSAR requirements specifications including the meta-model are released simultaneously. There are three types of AUTOSAR releases:

1. *Major release:* first digit change, contains backwards incompatible features.
2. *Minor release:* second digit change, contains backwards compatible features.
3. *Revision:* third digit change, contains bugfixes only.

4 Research Methodology

In order to provide the answer to the research question addressed in this study that is presented in the introduction, we developed a method named *SREA* that aims to reduce the costs and time associated with the process of analyzing the evolution of standardized requirements and evaluated it in a case study [12] of the AUTOSAR requirements, in collaboration with VCG.

We defined the SREA method by relying both on the existing and novel approaches. First, we conducted a literature review on the existing approaches for monitoring the evolution of system requirements using the *snowball* method [16]. In order to identify the starting set of papers, we searched for papers mentioning *requirements evolution* and *requirements volatility* keywords in their title, abstract and keywords sections using *Google Scholar*, *IEEE Xplore* and *Scopus* databases. We selected papers [2,9,13] as the starting point and continued to look for references and citation in these papers. We performed three iterations in total and analyzed in details 23 out of 42 relevant papers. Second, we improved our understanding of the case study context in semi-structured interviews, workshops and meetings with AUTOSAR experts from VCG. These two steps served as input for defining the SREA method.

Finally, we evaluated the method using AUTOSAR requirements as a unit of analysis from the chosen set of AUTOSAR releases. We chose all AUTOSAR releases from the latest major release *4.0.1* until the latest revision *4.2.2*. We decided not to consider previous AUTOSAR releases as their specification are not structured in the same way which makes them hard to be analyzed automatically. Nevertheless, AUTOSAR has a lot of significant changes in the last major release. Finally for the detailed analysis of changes between two releases, we decided to focus on the changes between *4.0.3* and *4.2.2* releases.

In order to be able to cope with the size of AUTOSAR that counts more than 21.000 requirements in its latest release (*4.2.2*), we developed a configurable software tool for gathering data and calculating and presenting the results to the AUTOSAR engineers at VCG. The tool compares different versions of the AUTOSAR requirements specifications in PDF and creates a structured report. The report presents the following information for the analyzed specifications: types of requirement changes in all analyzed releases, change history of each requirement and the number of requirements, cumulative number of requirements, number of changes and cumulative number of changes for each AUTOSAR release.

Finally in order to validate the proposed *SREA* method, we distributed a survey with 10 questions to six AUTOSAR experts at VCG. The questions were based on both quantitative and qualitative results of the method applied on AUTOSAR requirements, e.q., which AUTOSAR specifications are mostly unstable, and they aimed to assess whether the results of the method are in line with the expectation of the experts who participated in the development of the AUTOSAR standard. The experts were not aware of the method results.

5 The ***SREA*** method

The *SREA* method we propose in this paper consists of the following steps:

1. Define taxonomy of requirement changes in order to design the right metrics for performing the quantitative analysis in the next step.
2. Perform quantitative analysis of evolution of the requirements specifications in order to be able to correctly prioritize them in the next step.
3. Prioritize individual requirements specifications in order to select groups of specifications for the qualitative analysis in the next step.
4. Perform qualitative analysis of changes in order to accurately assess their impact on the system under development.

Step 1: Define taxonomy of changes: The first step of the *SREA* method aims to define the taxonomy of changes by:

1. Defining which types of changes shall be considered, e.g., added requirements, in order to define the metric for each type.
2. Defining the metrics for different types of changes, e.g., *NoA* as the number of added requirements, in order to calculate the total number of changes.

3. Defining the total number of changes, i.e., *NoC*, as the (weighted) sum of results of the previous metrics, in order to perform quantitative analysis.
4. Defining the taxonomy of modifications (which modifications shall be considered), e.g., requirements title, in order to perform qualitative analysis.

First three points aim to define the types of changes that shall be considered in the requirements evolution analysis, as not all changes have impact on the system under development (e.g., split requirements with low probability of occurrence). We specified the following metrics for each type of identified change:

1. *NoA* for the number of added requirements.
2. *NoS* for the number of split requirements.
3. *NoU* for the number of merged requirements.
4. *NoD* for the number of deleted requirements.
5. *NoM* for the number of modified requirements.
6. *NoC* for the total number of changed requirements.

The *NoA*, *NoS*, *NoU*, *NoD* and *NoM* are simple metrics that are calculated by counting the number of occurrences of each type of change. The *NoC* metrics is calculated as the sum of the results of other metrics, as shown in Formula 1:

$$NoC = a * NoA + b * NoS + c * NoU + d * NoD + e * NoM \tag{1}$$

The coefficients a, b, c, d and e are there to indicate which simple metrics shall be considered in the *NoC* metric, i.e., value 0 means that this particular type shall not be considered whilst value 1 means that it shall be considered.

The last point aims to define the taxonomy of requirements modifications that shall be considered in the analysis. Requirements that are added, split, merged and deleted are usually easily detectable based on their unique IDs or names. However, requirements that are modified require checking whether their content was changed. In practice, not all modifications to the content of the requirements are relevant, e.g., fixing spelling mistakes does not require effort for fulfilling the analyzed requirements. There are no general rules for deciding which modification are not relevant, so they have to be defined. Table 1 shows the taxonomy of the general types of modification we encountered in our study.

Additionally, it is advisable to implement the taxonomy of modifications in the tool responsible for calculating the metrics in a configurable way so that the inclusion/exclusion of each type could be done automatically.

Based on the defined taxonomy of changes, we can exclude from both quantitative and qualitative analysis all types of changes in the requirements that do not affect their semantics. This in turn is very valuable for the engineers analyzing the evolution of requirements as they are presented with the precise measure of requirements change considering only those requirements that actually require certain effort to be fulfilled by the system under development.

In order to successfully perform points 1–4 from the first step for a specific industrial case, a preliminary analysis of the requirements behavior shall be conducted. We propose the *micro* analysis method [14] that enables a comparison

Table 1. Taxonomy of modifications

Types of modifications	Description
Grammar and spelling corrections	Grammar and spelling improved in a new release.
Encoding modification	The specifications can be encoded in different ways and the output could slightly change.
Format modification	The format can change, e.g., how requirements are structured in tables or text.
Change in the technical term name	Changes in the name of, e.g., an API or a class.
Title modification	The title of a requirement can change.
Content modification	Modification in the content of a requirement.
Reference modification	Change of requirement's traceability reference

of two different versions of one significant (in terms of number of requirements and their scope) requirement specification for each category of requirements, e.g., functional requirements, design requirements, etc. This comparison aims to identify both different types of changes and different types of modifications. The outcome of the *micro* analysis should be a table with one row for each requirement changed and one column for each type of change encountered in the analysis, i.e., deleted, split, modified, merged and added requirements. Depending on the results of the *micro* analysis, we could then decide not to consider certain types of changes if their occurrence is insignificant for the analysis.

Step 2: Perform quantitative analysis of requirements evolution: In this step we aim to quantitatively analyze the evolution of standardized requirements in order to identify specifications that are mostly affected by changes. Our qualitative analysis relies on the *NoC* (Number of Changes) metric that counts the number of added, deleted, split, merged and modified requirements according to the defined taxonomy of changes. We analyze the evolution of requirement specifications in two ways: (i) by calculating the *NoC* and (ii) by considering the percentage of changes, based on the *RMI* metric that is derived from *NoC*.

The first one gives an overview of the amount of changes and which specifications contain the biggest number of changed requirements. The other one does that same taking also the total number of requirements into account. For example, although a specification with one thousand changes has a significant *NoC*, it could be quite stable, i.e., with a low percentage, if it contains ten thousands requirements. For this reason, we aim to assess the stability of each requirement by measuring *RMI* in the way defined in Formula 2.

$$RMI = \frac{R_t - NoC}{R_t} \tag{2}$$

Rt represents the total number of requirements for a specific version while *NoC* represents the total number of changed requirements between this version and the previous one. Substructing the results of the *RMI* metric from *Rt* can be used for calculating the percentage of changed requirements. Note that the percentage could exceed 100 % in some cases because *RMI* considers all types of changes including merged and deleted requirements. For example, one specification could have 199 *NoD*, 20 *NoA*, and 47 *NoM*, hence 266 *NoC* from one version to another. However it could have just 200 requirements in the last version.

Step 3: Prioritize individual requirements specifications: The third step of the proposed method aims to collect a group of specifications based on the results of the previous step and the importance of each specification for the system under development. One specification is usually considered important if its requirements are needed for adopting a specific standardized feature in the system. The set of prioritized specifications are then grouped according to their semantics (e.g., relevant standardized features) in order to serve as input to the next step of the method. We do not specify the number of specifications that should be prioritized and grouped because it depends on the needs, e.g., adopting one small standardized feature can affect only a few requirements specifications while adopting an entire new release of a standard may affect many.

Step 4: Perform qualitative analysis of changes: The last step of the *SREA* method is focused on the qualitative analysis of changes in the prioritized group of requirement specifications. The analysis of the actual changes in the requirements is done by comparing their content (i.e., whether their textual representation is the same or not) between different releases of the standard. The outcome of this step is a report for each prioritized specification or for a group of specifications related to the analyzed feature. In order to increase its readability, we propose to structure the report in the following way:

1. *Table of Contents* contains a list of sections and subsections of the report.
2. *General Data* contains the results of the *NoC* metric, for each type of change, and *RMI* metrics, calculated again for the prioritized specifications.
3. *List of Changed Requirements* contains the list of all types of changes considered by specifying the ReqId, title and content of each requirement.
4. *Detail of Modified Requirements* contains the comparison between the content of all modified requirements emphasizing (e.g., bold or coloring) the modified text according to the taxonomy defined in step 1.

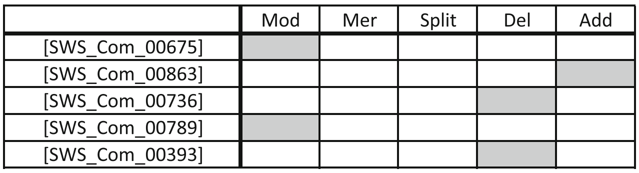
Since the main goal of *SREA* is to increase the speed of analysis of standardized requirements evolution, automated tool support for performing both quantitative and qualitative analysis described in the steps above is an important part of the method. This tool should also be able to generate the final report based on the configured taxonomy of changes, as already explained.

6 Evaluation of the *SREA* Method on AUTOSAR

In this section, we show partial results from the evaluation of the *SREA* method on the evolution of AUTOSAR requirements for a specific objective: to facilitate updates of the AUTOSAR modeling tools with new releases of AUTOSAR.

This implies focusing on the analysis of AUTOSAR specifications containing design requirements. We organize this section according to the steps of the method.

Step 1: Define taxonomy of changes: We initially performed *micro* analysis using two AUTOSAR specifications: *AUTOSAR_TPS_SystemTemplate*, containing design requirements, and *AUTOSAR_SWS_Com*, containing BSW requirements. Figure 1 shows an extract of the results for *SWS_COM*.



	Mod	Mer	Split	Del	Add
[SWS_Com_00675]	▨				
[SWS_Com_00863]					▨
[SWS_Com_00736]				▨	
[SWS_Com_00789]	▨				
[SWS_Com_00393]				▨	

Fig. 1. Short extract of the *micro* analysis

Based on the complete results of the *micro* analysis of the *SWS_COM* and *TPS_SystemTemplate* specifications, we concluded that the number of merged and split requirements is very low and therefore insignificant for our study. Therefore, we decided not to consider split and merged requirements because of their low probability of appearance and increased difficulty in detection. The decision not to consider these types of requirement changes resulted in the definition of the *NoC* metric we used for AUTOSAR evaluation presented in Formula 3.

$$NoC = 1 * NoA + 0 * NoS + 0 * NoU + 1 * NoD + 1 * NoM \tag{3}$$

Based on the taxonomy of modifications, and in the discussion with the engineers from VCG, we decided to consider only the following types of modification for the quantitative analysis performed in step 2: *Change in the technical term name*, *Title modification* and *Content modification*.

Step 2: Perform quantitative analysis of requirements evolution: The results of the quantitative analysis of the AUTOSAR requirements evolution are presented in Fig. 2. We considered all minor releases and revisions of the AUTOSAR major release 4 (i.e., releases from *4.0.1* to *4.2.2*).

We can observe the evolution of the AUTOSAR standard in Fig. 2 from two different perspectives: (i) the *NoC* across consequently releases (left chart) and (ii) the total number of requirements (right chart). Based on these charts, we can see that AUTOSAR is continuously changing through its releases. Furthermore, we can also see that AUTOSAR is continuously *growing*: In *R401* AUTOSAR had 14.000 requirements whilst in the last version (*R422*) it counts more than 21.000 requirements. Finally, we can see that minor releases of AUTOSAR (*R411* and *R421*) bring more changes to the requirements than revisions.

In order to analyze the specifications that are mostly affected by the evolution of the AUTOSAR standard, we sorted them based on the results of the *NoC* and

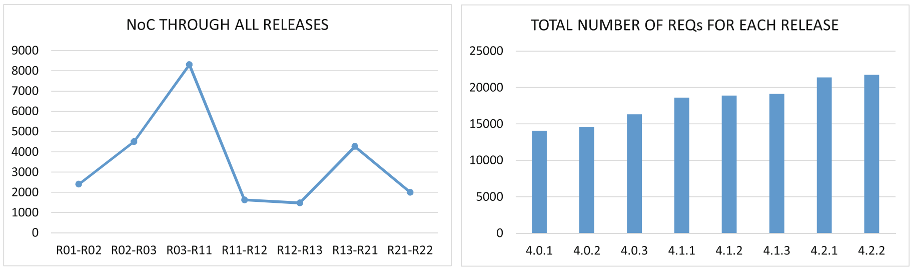


Fig. 2. Results of the quantitative analysis of the AUTOSAR requirements evolution

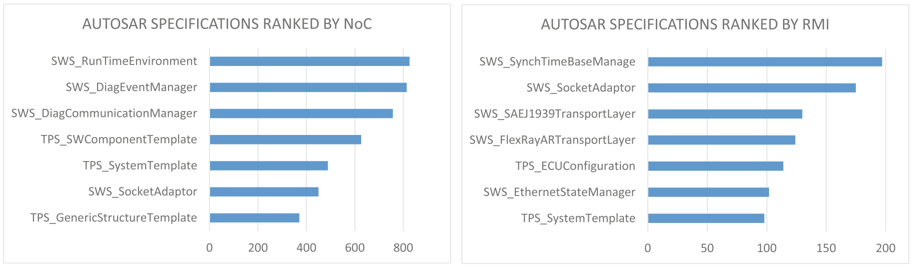


Fig. 3. AUTOSAR specifications ranked according to the *NoC* and *RMI*

RMI metrics calculated between the chosen AUTOSAR releases *4.0.3* and *4.2.2*. Figure 3 shows the first 7 (over 84) specifications.

The results of the *NoC* metric show that *SWS_RunTimeEnvironment* and *SWS_DiagEventManager* specifications are mostly affected by the changes (more than 800 changes). Although they have a significant *NoC*, we can not directly conclude that these specifications are also the most unstable ones. To assess this, we ranked them by the results of the *RMI* metric in order to investigate the relations between the two lists. We can see that *SWS_SynchTimeBaseManager* and *SWS_SocketAdaptor* specifications have the highest *RMI* value. By combining the two lists, we can also see that *SWS_SocketAdaptor* and *SWS_SystemTemplate* are the only two specifications that are considered highly affected by the evolution based on the results of both metrics. Generally, all the specification that have high values of *NoC* and *RMI* are good candidates to be analyzed first, depending on their importance for the actual system under development.

Step 3: Prioritize individual requirements specifications: In this step, we focused on the main objective of this analysis: facilitate the updates of the AUTOSAR modeling tools based on a new release of AUTOSAR. Therefore, we asked the AUTOSAR engineers from VCG which specifications are considered the most important for the analysis of impact on the AUTOSAR modeling tools. They agreed on the following three specifications: *TPS_SystemTemplate*, *TPS_SWComponentTemplate* and *TPS_ECUConfiguration*.

Based on this prioritization and the outcome of the previous step that identified specifications mostly affected by the changes, we decided to focus on the qualitative analysis of *TPS_SystemTemplate* and *TPS_SWCTemplate* in the first phase. The *TPS_SystemTemplate* specification contains general design requirements on how the system shall be designed, e.g., description of ECUs connected with electronic buses and transmission of signal on the electronic buses. The *TPS_SWComponentTemplate* specification contains general design requirements on how software components should be designed with their data interfaces.

Step 4: Perform qualitative analysis of changes: In the last step we performed qualitative analysis on the prioritized specifications by running the tool we implemented and by providing a detailed report. In this report, we first show the results of all proposed metrics, as presented in Table 2.

Table 2. Report - results of the metrics applied on the chosen specifications

Metric	TPS_SystemTemplate	TPS_SWCTemplate	Total
NoA	472	374	846
NoD	4	24	28
NoM	13	228	241
NoC	489	626	1115
RMI	0,01	0,41	0,28

The only significant difference in results between the two specifications is for the *NoM* metric. *TPS_SystemTemplate* counts only 13 modifications whilst *TPS_SWComponentTemplate* counts 228 modifications. Nevertheless, the *RMI* metric indicates higher stability of the *TPS_SWComponentTemplate* (0,41) in comparison to the *TPS_SystemTemplate* (0,01). This is because the number of requirements in the last version of the *TPS_SWComponentTemplate* is much higher than in the last version of the *TPS_SystemTemplate* (1069 compared to 494) with much smaller difference in the *NoC* value (626 compared to 489).

After showing the results of all metrics, we list in the report all added, removed and modified requirements emphasizing the textual modifications in the modified requirements in bold. The example of the presentation of one modified specification items in the report is shown in Fig. 4.

One interesting discovery that we came across by analyzing changes in the AUTOSAR requirements was the fact that change in the name of certain technical terms (e.g., API or meta-class names) can have a significant impact on the results of the metrics. For example, we located an unexpected increase in the results of the *NoM* metric between AUTOSAR releases *4.2.1* and *4.2.2*. After investigating the causes of this, we found that AUTOSAR renamed one requirement specification from *SWS_DevelopmentErrorTracer* to *SWS_DefaultErrorTracer*. As a consequence of this renaming, all requirements that contained the word *development* (in this context) have also been renamed to *default*.

5.54 [TPS_SWCT_01209]

———content———

previous version:
ClientServerAnnotation

the clientserverannotation can be used to provide more information with respect to the operation of the port.

current version
ClientServerAnnotation

the clientserverannotation can be used to provide more information with respect to the **clientserveroperation** of the **portprototype.**

————————————

Fig. 4. Example of modification of a specification item

In order not to invalidate the results by this change that has no impact on the semantics of the analyzed requirements, we provided an option in our tool for the engineers to specify which changes in the names of the technical terms shall be ignored (e.g., every change of a single word from *development* to *default*). Using this option, engineers can add these types of modifications to the configuration file every time they encounter them and run the tool again. We discovered that ignoring these types of replacements can significantly decrease the total *NoM* and therefore the size of the report providing more accurate results to the users. For example, excluding the *development* to *default* replacement reduced the total *NoM* between the two analyzed releases by 27 %.

7 Discussion

We validated the proposed *SREA* method by distributing a survey to the AUTOSAR experts at VCG in which we asked them a number of questions related to the evolution of the AUTOSAR requirements specifications, e.g., which specifications they think are mostly affected by the changes between two AUTOSAR releases. Our goal was to assess whether the results of our method meet their expectations. We concluded that the results of the *SREA* method fully met the answers from the VCG experts in 66 % of questions and were significantly different in just 1 %. In 17 % of questions we did not get an answer from the experts and in 16 % of questions the answer was slightly different than the one provided by our method, e.g., the experts indicated that the second most affected document according to our method was mostly affected by the changes. More details about the validation including survey questions and answer can be found in [10].

Based on the results of the validation, we concluded that the proposed method can indeed be used for analyzing the evolution of standardized requirements of AUTOSAR and identifying the requirements specifications, together with the actual requirements, that are mostly affected by the changes. However, additional validation of the true benefits of the proposed method and the tool in reducing the amount of time spent analyzing the evolution of standardized system requirements is yet to be performed, as we explained in the future work.

In order to assess different threats to validity for our study, we followed Cook and Campbell's list of threats [5], i.e., threats to internal, external, construct and conclusion validity. Due to space limitation, we describe in this paper the most important threats to internal validity, that concerns accuracy of our results, and external validity, that concerns generalization of our results.

The most important threat to the internal validity of our study is related to what is considered a requirement in the AUTOSAR specifications. According to AUTOSAR, not only specification items and constraints described in this paper can be considered as requirements, but also plain text written in the specifications as it is mandatory to be followed when developing AUTOSAR compliant systems. However, we realized during our micro analysis that most of the important statements are part of specification items (and constraints), whilst the remaining text usually represents examples, rationales, and figures. For this reason, we believe that the internal validity of our results is still high as we managed to analyze the most important content of the requirements.

The most important threat to the external validity of our study is related to the generalizability of our results to systems that are developed based on other industrial standards and their requirements. Although we cannot claim that the *SREA* method can provide equally good results in other domain without evaluating it in additional case studies, we believe that the this is likely due to the fact that we designed the steps of the method considering the existing literature and studies performed and validated in different domains (e.g., avionics).

Finally, we can recommend to other companies who would like to analyze the evolution of standardized system requirements to start by defining the taxonomy of possible requirements modifications and the types of changes according to their knowledge about the analyzed standard. We believe that the other steps of the method are applicable to other contexts/domains as well.

8 Conclusion

In this paper, we present and evaluate the method named *SREA* that can be used to facilitate the process of adopting new releases of industrial standards and their features. The method is based on the quantitative and qualitative analysis of evolution of the standardized system requirements and is able to:

- Identify, based on the *NoC* metric, requirements specifications that are mostly changed in the new release of a standard, indicating that they should be considered first in the analysis of impact of adopting the new release.

– Identify, based on the *RMI* metric, requirements specifications that are mostly unstable during the evolution of one standard, indicating that features described in these specifications may contain defects.
– Present the actual content of added/removed and modified requirements in the concrete specifications of one release of a standard to the engineers performing the analysis, thus significantly reducing the time of analysis.

We apply and validate the *SREA* method on the case of AUTOSAR standard by developing the software tool that implements the method. We used the proposed method and the tool to study and assess the impact of AUTOSAR requirements evolution on the automotive software systems based on AUTOSAR. Our results show that the requirements standardized by AUTOSAR and their evolution should be analyzed and measured in a structured and automated way, i.e., by following a clearly defined method supported by a software tool to automate the process of gathering results. This approach helped automotive engineers from Volvo Car Group to faster assess the impact of AUTOSAR design requirements changes, related to a set of new AUTOSAR features, on the AUTOSAR modeling tools used in the development process.

In particular, we show that by applying the *SREA* method to different versions of the AUTOSAR standard, it is possible to identify the most important requirements specifications to be analyzed in the first phase. As an example, we showed the analysis of the two specifications - *TPS_SystemTemplate* and *TPS_SWComponentTemplate* - shall be done first in order to assess the impact of switching from the AUTOSAR release *4.0.3* to release *4.2.2* on the used AUTOSAR modeling tools. We also show that the method is able to provide a report containing only relevant information on the added, removed and modified requirements to the automotive engineers in order to increase the speed of analysis. For example in case of the *TPS_SystemTemplate*, the report contained 269 pages of relevant information about the changes whilst the same specification provided by AUTOSAR have around 1500 pages. Without the proposed method, these pages would need to be compared manually between the analyzed releases.

The information provided by the *SREA* method can therefore help organizations responsible for managing large software systems in understanding which areas of the system will be mostly affected by the changes in the standardized requirements and therefore faster adopt new releases of the standard.

We identified several potential areas of interest for further work. Since requirements evolution is today considered to be a challenging task for both industry and academia, this study shall be considered as a first step in defining the methodology for performing this task. There are several interesting ways for improving and/or extending the method we propose, in particular:

1. Calculate the actual engineering effort that is saved by using the *SREA* method. This would increase the validity of the presented results.
2. Apply *SREA* to other industries that develop system based on standards. This would increase the generalizability of the presented results.

3. Extend the proposed method to provide effort estimates for adopting new standardized features in the development projects. This would additionally help companies in allocating resources for supporting specific features.
4. Extend the proposed method to include a model for estimating the number of changes to the requirements specifications that will occur in the future releases of the standard. This would help standardization organizations in allocating resources for working with the most critical specifications early.

The tool we used for the analysis of AUTOSAR requirements can be downloaded from here: https://www.chalmers.se/en/projects/Documents/SREA.zip

Acknowledgment. The authors would like to thank Swedish Governmental Agency for Innovation Systems (VINNOVA) for funding this research (grant no. 2013-02630) and the AUTOSAR team at Volvo Car Group for contributing to the work.

References

1. Anderson, S., Felici, M.: Controlling requirements evolution: an avionics case study. In: Koornneef, F., Meulen, M. (eds.) SAFECOMP 2000. LNCS, vol. 1943, pp. 361–370. Springer, Heidelberg (2000). doi:10.1007/3-540-40891-6_31
2. Anderson, S., Felici, M.: Requirements evolution from process to product oriented management. In: Bomarius, F., Komi-Sirviö, S. (eds.) PROFES 2001. LNCS, vol. 2188, pp. 27–41. Springer, Heidelberg (2001). doi:10.1007/3-540-44813-6_6
3. AUTOSAR, www.autosar.org: Automotive Open System Architecture (2003)
4. Broy, M., Kruger, I., Pretschner, A., Salzmann, C.: Engineering automotive software. Proc. IEEE **95**(2), 356 (2007)
5. Cook, T., Campbell, D.: Quasi-Experimentation: Design & Analysis Issues for Field Settings. Houghton Mifflin, Boston (1979)
6. Durisic, D., Staron, M., Tichy, M.: ARCA - Automated analysis of AUTOSAR meta-model changes. In: International Workshop on Modelling in Software Engineering (2015)
7. Durisic, D., Staron, M., Tichy, M., Hansson, J.: Evolution of long-term industrial meta-models - a case study of AUTOSAR. In: Euromicro Conference on Software Engineering and Advanced Applications, pp. 141–148 (2014)
8. Ernst, N., Borgida, A., Jureta, J., Mylopoulos, J.: An overview of requirements evolution. In: Mens, T., Serebrenik, A., Cleve, A. (eds.) Evolving Software Systems, pp. 3–32. Springer, Heidelberg (2014)
9. Li, J., Zhang, H., Zhu, L., Jeffery, R., Wang, Q., Li, M.: Preliminary results of a systematic review on requirements evolution. In: Proceedings of the IEEE Conference on Evaluation Assessment in Software Engineering, pp. 12–21 (2012)
10. Motta, C.: Analyzing the Evolution of System Requirements. Chalmers — University of Gothenburg (2016)
11. Nurmuliani, N., Zowghi, D., Fowell, S.: Analysis of requirements volatility during software development life cycle. In: Proceedings of the Australian Software Engineering Conference, pp. 28–37 (2004)
12. Runeson, P., Host, M.: Guidelines for conducting and reporting case study research in software engineering. In: Proceedings of the Conference on Empirical Software Engineering, pp. 131–164 (2009)

13. Shi, L., Wang, Q., Li, M.: Learning from evolution history to predict future requirement changes. In: Proceedings of the International Conference on Requirements Engineering, pp. 135–144 (2013)
14. Stark, G., Skillicorn, A., Smeele, R.: A micro and macro based examination of the effects of requirements changes on aerospace software maintenance. In: Proceedings of the IEEE Conference on Aerospace, pp. 165–172 (1998)
15. Wang, H., Li, J., Wang, Q., Wang, Y.: Quantitative analysis of requirements evolution across multiple versions of an industrial software product. In: Proceedings of the 17th Conference on Asia-Pacific Software Engineering, pp. 43–49 (2010)
16. Wohlin, C.: Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering (2014)