Abstract—Meta-models in software engineering are used to define properties of models. Therefore the evolution of the meta-models influences the evolution of the models and the software instantiated from them. The evolution of the meta-models is particularly problematic if the software has to instantiate two versions of the same meta-model - a situation common for long-term software development projects such as car development projects. In this paper, we present a case study of the evolution of the standardized meta-model used in the development of the automotive software systems – the AUTOSAR meta-model – at Volvo Car Corporation. The objective of this study is to assist automotive software designers in planning long term development projects based on multiple AUTOSAR meta-model versions. We achieve this by performing quantitative analysis of the AUTOSAR meta-model evolution in order to visualize the size and the complexity change between different meta-model versions and calculate the number of changes which need to be implemented to adopt a newer version. The analysis is done for each major role in the automotive development process affected by the changes.

I. INTRODUCTION

The evolution of software today is influenced by the evolution of models and also meta-models. The meta-models are used to define the properties of the models and as such they influence the software instantiated from these models [1]. We consider a model as an abstract representation of a software system and a meta-model as a model which defines the syntax and the semantics of a particular domain-specific modeling environment [2], [3]. One example of such domain-specific meta-model used in industry is the standardized meta-model used in the development of automotive software systems - AUTOSAR (AUTomotive Open System ARchitecture) [4] meta-model. A simplified example of the usage of the AUTOSAR meta-model to allocate software components onto different Electronic Control Units (ECUs)\(^1\) is presented in Figure 1.

As industrial models, like AUTOSAR, are usually exchanged between a number of stakeholders in the development process which may use different tools, meta-models are used as basis for the development of these tools in order to assure tooling interoperability. Therefore the compliance of the models to their meta-models must be preserved to enable different tools to work on the same models. For this reason, the evolution of the meta-models is very important to provide means to express new modeling solutions and as such enable innovation in the software based on these solutions.

\(^1\)Embedded system (hardware and software) responsible for one or more vehicle functions (e.g. engine control, breaking).
extracting the data from different versions of the AUTOSAR meta-model. Finally we perform quantitative analysis of the data by applying a number of metrics to visualize the size and the complexity trends in the evolution and counting the number of changes between different meta-model versions.

The rest of the paper is structured in the following way: Section 2 describes the case study context - automotive software development based on the AUTOSAR standard. Section 3 describes the related work. Section 4 describes the research goal and the research questions and presents the design of the case study. Section 5 presents the results of the case study. Section 6 discusses and validates the results of the study and provides recommendation to other companies for monitoring the evolution of their meta-models. Finally, Section 7 summarizes our conclusions and plans for future work.

II. Case Study Context

Automotive software systems are distributed systems where one premium vehicle today typically contains 70 - 100 ECUs [6]. Together with their distributed nature, the development of the automotive software systems is also distributed as they are developed in a collaborative environment which involves a number of stakeholders. On one side we have car manufacturers (OEMs - Original Equipment Manufacturers) responsible for designing and verifying the functions and the architecture of the system. On the other side we have different layers of suppliers (e.g. application software suppliers, tool suppliers, hardware suppliers) responsible for design, implementation and verification of the specific components in the system. In addition to the high complexity implied by the distributed development, the complexity of the automotive software systems is constantly increasing [7] due to new features in cars [8].

In order to facilitate the distributed development of automotive software systems, AUTOSAR standard has been introduced with the goal to separate the responsibilities of different stakeholders in the process. This separation is based on a three layer software architecture which aims to separate the application software from the underlying basic software (signaling, network management, diagnostics, etc.). Based on this architecture, AUTOSAR provides standardized interfaces between architectural components in order to standardize the exchange format for their models. The models are expressed using XML and the XML schema used for the validation by the AUTOSAR based tools is generated from the AUTOSAR meta-model [9]. A simplified sketch of the AUTOSAR software development process in presented in Figure 2.

The AUTOSAR meta-model hierarchy is based on the Meta-Object Facility (MOF) standard [10] and it contains 5 meta-layers (4 meta-layers plus MOF). The difference between the classical MOF meta-layers (MOF Mx) and 5 AUTOSAR meta-layers (AR Mx) is that AUTOSAR defines classifiers and their instances (objects) on two different layers while according to MOF they are both defined on MOF M1 (dual classification problem, see [11]). This is depicted in Figure 3.

![Fig. 3. AUTOSAR - MOF relation](image-url)

AR M3 (AUTOSAR Profile meta-layer) is based on the UML 2.0 and it defines the used UML stereotypes and annotations. AR M2 (AUTOSAR Templates meta-layer) defines how to design the automotive electrical system (ECUs, software components, etc.). AR M1 (AUTOSAR User Models meta-layer) represents the actual models developed by the system designers. Finally AR M0 (AUTOSAR User Objects meta-layer) represents the realization of the AUTOSAR models in the actual ECU. In this paper, we analyze the evolution of the AR M2 and the standardized part of the AR M1 meta-layers.

The AR M2 meta-layer consists of a hierarchy of classifiers with their attributes and it is divided into different AUTOSAR templates. Each template is used to define how to model one specific part of the automotive system (e.g. Software Component template defines software components and their interaction, System template defines communication between ECUs, etc.). The AR M1 meta-layer consists of instances of the AR M2. The instances used for modeling ECU application software are developed by the software designers while the instances used for modeling the configuration of ECU basic software are standardized by AUTOSAR (e.g., COM stack responsible for the ECU communication, I/O responsible for the access to sensors and actuators, Services such as Diagnostics, etc.). In the analysis of the AR M1 evolution, we consider only the standardized part (models of the ECU configuration).

AUTOSAR uses a three digit numbering scheme Rx.y.z to identify different releases which all include a new release of the meta-model. The first digit identifies major releases which are not compatible between each other and should be considered independently. The second digit identifies minor releases which include compatible extensions and bug-fixes. The third digit identifies revisions which usually contain bug-fixes only. The first two digits identify one evolution branch. Maximum two branches may be maintained by the AUTOSAR consortium in parallel where one branch represents a Development branch focused on bug-fixes and innovations, and the other branch represents a Maintenance branch focused mostly on bug-fixes.
III. RELATED WORK

There is a lot of research today related to the visualization of the software evolution, as presented in the systematic mapping study by Novais et al. [12]. For example Lanza et al. [13] use several object-oriented metrics for visualizing the evolution of classes like us, however they focus on the evolution of source code. Some of the papers are also related to the visualization of the model and meta-model evolution such as the one from Madhavi et al. [14] who propose a framework for visualizing model-driven software evolution or the one from Lange et al. [15] who propose a tool to aid users in tasks such as model understanding, identification of quality problems and evolution trends. However, these papers are analyzing the evolution of entire models without considering their specific parts relevant for different roles. There is also a lack of empirical research in this area, especially related to the visualization of large scale meta-model evolution.

Many papers also present different methods for mining software repositories in the context of the software evolution, as presented by Kagdi [16]. For example Zimmermann et al. [17] and Ying et al. [18] build prediction models to predict which classes, functions and attributes will be changed based on the historical analysis of different source code versions. With respect to meta-model evolution, Vermolen et al. [19] present an interesting research about the coupled evolution of meta-models and models. They propose a method for detecting and formalizing the complex meta-model evolution in order to migrate the existing models according to the new meta-models more easily. However we believe the area of meta-model evolution can also be improved with more empirical studies, especially related to the validation of the proposed methods for re-constructing and monitoring the meta-model evolution on industrial meta-models.

IV. CASE STUDY DESIGN

We conduct a case study analysis of the AUTOSAR meta-model evolution at VCC based on the guidelines from Kitchenham et al. [20] and Runeson et al. [21]. Our research objective is defined according to the structure presented by Wohlin et al. [22] as:

- **Goal:** Analyze the AUTOSAR meta-model evolution.
- **Purpose:** Assist software designers in assessing the size and complexity increase between AUTOSAR meta-model releases and the number of changes to be implemented for adopting a new release.
- **View:** Software designers working with models instantiating multiple AUTOSAR meta-model releases.
- **Context:** Automotive embedded software systems based on the AUTOSAR standard.

In order to achieve this objective, we define the following research questions:

- **Q1:** What is the trend in the size change between AUTOSAR meta-model releases?
- **Q2:** What is the trend in the complexity change between AUTOSAR meta-model releases?
- **Q3:** How many changes need to be implemented to adopt a new AUTOSAR meta-model release?
- **Q4:** Which roles are mostly affected by the evolution of the AUTOSAR meta-model?

In order to provide answers to the research questions, we design our case study analysis around the following 5 steps:

**Step A:** In order to identify the most important roles in the development process which are affected by the AUTOSAR meta-model evolution.
**Step B:** Map the identified roles to the relevant parts of the AUTOSAR meta-model.
**Step C:** Identify the relevant types of changes to be considered in the AUTOSAR meta-model evolution.
**Step D:** Define which AUTOSAR meta-model releases shall be considered and extract the relevant data from them.
**Step E:** Calculate the metrics on each considered release and visualize their results.

In the workshop mentioned in step B, we agreed upon the relevant types of changes to be considered in the analysis based on the analysis of a small sample of changes. We define 'relevant' changes as changes which require certain implementation and/or integration effort.

**Step D:** In the workshop mentioned in steps B and C, we agreed upon the set of AUTOSAR meta-model releases which shall be considered in the analysis. We used a meta-data model presented in Figure 4 for the extraction of the relevant data from the considered releases. The meta-data model is based on the relevant part of the MOF meta-model.

**Fig. 4.** Meta-data model used for the measurements

Meta-models are divided into Packages which contain Elements - classifiers and instances. Classifier Elements contain...
Analyze the AUTOSAR meta-model evolution

Q1: What is the trend in the size change between meta-model releases?
Q3: How many changes need to be implemented to adopt a new AUTOSAR meta-model release?
Q2: What is the trend in the complexity change between meta-model releases?

Based on the presented meta-data model, we developed a tool to extract the relevant data from the AR M2 and the AR M1 meta-layers designed in the Enterprise Architect tool (used by AUTOSAR meta-model developers). Due to the structure of the AUTOSAR meta-model, Elements in the AR M2 represent classifiers and Elements in the AR M1 represent instances.

Step E: In order to measure the properties of the AUTOSAR meta-model evolution, we used the metrics presented in Figure 5 driven by the Goal-Question-Metric approach [23]. The chosen metrics are based on the structural object-oriented metrics defined by Genero et. al. [24] and Yi et. al. [25].

The Number of elements (NoE) and the Number of attributes (NoA) metrics count the total number of Elements / Attributes respectively in each meta-model release. We use these metrics to measure the increase in size of the meta-model during its evolution as elements (classifiers with their attributes and objects) represent the main building blocks of the AUTOSAR meta-model.

The Number of changes (NoC) metric counts the total number modifications, additions or removals of the meta-elements of the meta-data model. This means that in case one Attribute changed both its Name and its Type, this counts as two changes. Additionally when introducing or removing meta-elements containing other meta-elements (e.g., Elements with Attributes and Connectors), the total number of changes is defined as the total number of modifiable meta-elements contained in the introduced / removed meta-element plus one for the introduced / removed meta-elements itself. For example if one Attribute with three Annotations is removed, this counts as four different changes - one for the Attribute and three for the Annotations. This behavior is justified by the fact that introduction of one Element cannot be counted as one change, like for example a change of Connector’s lower bound, as it requires much more effort to be implemented. The Number of changed elements (NoCE) and the Number of changed attributes (NoCA) metrics are based on the NoC metric but this time, all modifications, additions and removals of one Element and Attribute respectively count as one change only. We use these metrics to identify the roles which are mostly affected by the changes and to count the number of changes needed to be implemented to adopt a new AUTOSAR meta-model release.

The Complexity (CPX) metric represents a sum of Henry and Kafura’s structural complexities [26] of all Elements in one meta-model release. It is defined as

$$\text{CPX}(n) = \sum_{i=1}^{n} [\text{FanIn}(i) \times \text{FanOut}(i)]^2$$

where $n$ represents a number of Elements and $\text{FanIn}(i)$ / $\text{FanOut}(i)$ a number of sourceConnectors / destinationConnectors (not counting Connectors of Type Generalization) of the Element $i$ respectively. Generally metrics based on fan-in and fan-out are widely used for measuring the structural complexity of modules [7]. Fan-in is defined as the number of modules which are calling a given module while fan-out is defined as the number of modules which are called by the given module. As modules in the AUTOSAR meta-model represent Elements connected by Connectors, it is not possible to call one module from another. However since different Elements may be part of different domains and as such modeled by different teams, any interaction between them can be considered as increase in the overall complexity of the AUTOSAR meta-model. Therefore we consider a sourceConnector as fan-out and a destinationConnector as fan-in property of the Element rather than just its Attribute used for the size measurement.

Finally the Average depth of inheritance (ADIT) metric calculates the average number of parent Elements (connected by Generalization Type of Connectors) for all Elements in one AUTOSAR meta-model release and it complements the CPX metric in measuring the complexity increase between releases.

In order to calculate the metrics on the extracted data from each considered release of the AUTOSAR meta-model, we developed a tool to compare the models of different releases which is also able to visualize the results using line charts, histograms and heatmaps. As Elements in the AR M1 meta-layer represent instances with no Attributes nor Connectors of type other than aggregation (containers aggregating parameters), the NoA, NoCA, CPX and ADIT metrics are not applicable to this meta-layer.

V. CASE STUDY RESULTS

In this section, we present the results of the case study structured according to the steps in the case study design.

A. Identified roles

Based on the interviews with software engineers from VCC, we identified the following roles in the AUTOSAR based
software development process (our objective was to capture the most important roles but other roles may exist too):

1) **Application software designers** - a team at the OEMs responsible for designing vehicle functions by defining software components and their interaction.

2) **ECU communication designers** - a team at the OEMs responsible for designing the communication between ECUs (e.g., transmitting signals on buses).

3) **ECU basic software configurators** - a team at the OEMs responsible for specifying the basic software configuration (i.e., which parameters are needed).

4) **Basic software designers** - a team at the basic software suppliers responsible for designing the basic software modules (e.g., interfaces, services, etc.).

5) **ECU communication configurators** - a team at the application software suppliers responsible for configuring ECU communication basic software modules.

6) **Diagnostics configurators** - a team at the application software suppliers responsible for configuring car diagnostics basic software modules.

7) **Upstream mapping tool developers** - a team at the tool suppliers responsible for automated derivation of parts of the ECU configuration from the models.

### B. Mapping of roles

In a workshop with the AUTOSAR team from VCC, we mapped the identified roles to the relevant parts of the AUTOSAR meta-model (if they are affected by the changes in these parts). The outcome is presented in Figure 6 (‘X’ denotes that the corresponding role is affected by the changes in the corresponding part of the meta-model). We use this mapping to analyze the results for the identified roles separately.

![Fig. 6. Mapping of roles to meta-model parts](image-url)

We identified that the mapping of roles to meta-model parts is not 1:1. This means that several roles may be affected by the changes in the same part of the AUTOSAR meta-model and also that changes in different parts may affect the same role. We also identified that not all parts of the meta-model are covered by the identified roles, in particular the **Methodology**, the **EcuResourceTemplate** and non-communication and non-diagnostic parts of the **AR M1**. As the **Methodology** part is auxiliary, the **EcuResourceTemplate** is not currently used and other non-communication and non-diagnostic parts of the **AR M1** are relevant only to specific roles which do not have a significant impact on the development process, we decided to exclude them from the analysis even though they may be relevant for some additional roles.

### C. Relevant types of changes

By examining a small sample of AUTOSAR meta-model changes between R4.0.3 and R4.1.1 with the AUTOSAR team at VCC, we realized that several types of changes which belong to the relevant parts of the meta-model are not relevant for any of the mapped roles. We considered a change not to be relevant if it does not require any implementation / integration effort, such as editorial changes in the *Notes of Elements / Attributes* or change in the format of the **Annotations**. Therefore we decided to exclude from the analysis the changes to the *Notes of Elements / Attributes / Connectors*. We also decided to consider only the changes to **Annotations** (i) of type ‘obsolete’ (the **Element** will be removed in future), (ii) related to configuration classes of **AR M1** instances (when the instance shall be defined, i.e. pre-compile time, link time or post-build time) and (iii) related to the identification of the model **Elements** (**AR M2**) from which the ECU configuration **Elements** (**AR M1**) are derived from.

In order to validate our assumption that it is necessary to exclude the changes which are not relevant from the analysis, we compared the results of several metrics considering all and considering the relevant changes only. Figure 7 shows an example of the comparison between the number of all changes and the number of relevant changes in **R4** (i.e. releases R4.x.y).

![Fig. 7. Number of all vs. relevant only changes in R4](image-url)

We can see that the number of all changes between **R4.0.2** and **R4.0.3** was less than the number of all changes between **R4.0.1** and **R4.0.2** even though the number of relevant changes only between **R4.0.2** and **R4.0.3** was increased in comparison to the number of relevant changes only between **R4.0.1** and **R4.0.2**. This behavior is explained by many editorial (not relevant) changes between **R4.0.1** and **R4.0.2**. There are several other similar cases like this so we concluded that the results considering all and considering relevant changes only differ quite a lot. This result validates our assumption that wrong conclusions can be derived from the results considering all changes in the AUTOSAR meta-model.
D. Considered releases

In a workshop with the AUTOSAR team at VCC, we agreed to consider only the AUTOSAR meta-model releases presented in green in Figure 8. Apart from these releases, there are three additional release branches in the beginning of AUTOSAR (R1.0, R2.0 and R2.1) which we decided not to consider for two reasons. First, they are not used today. Second, the release process back then was quite different (releases occurred after every change or a small group of changes), plus the maturity of the meta-model was not as good as in R3.0.1 and onwards. Additionally we decided not to consider releases R3.0.4 - R3.0.7 as their maintenance was negligible due to the fact that most of the AUTOSAR partners quickly moved to release branch R3.1.

E. Measurement results

In this section, we present and analyze the results of the measurements applied on the considered set of AUTOSAR meta-model releases with respect to the research questions.

**Q1: Size trend:** In order to measure the trend in the size increase between different releases of the AUTOSAR meta-model, we compare the results of the NoE and the NoA metrics for all considered releases. Figure 9 shows the number of relevant Elements per each minor release / revision in R3 (i.e. releases R3.x.y) and R4 (i.e. releases R4.x.y).

![Number of elements - R3 and R4](image)

We can see a much higher increase in size between different major releases (R3 and R4) and also between minor releases (e.g. R3.1.5 to R3.2.1 and R4.0.3 to R4.1.1) in comparison to revisions. Similar results can be seen by comparing the number of Attributes between different releases which is also true for the analysis of the identified roles separately.

**Q2: Complexity trend:** In order to measure the trend in the complexity increase between different releases of the AUTOSAR meta-model, we compare the results of the ADIT and the CPX metrics for all considered releases. The results of the ADIT metric for different roles in R3 are stable and the same is true for the results of the CPX metric (except a small decrease between R3.0.1 and R3.0.2 and a small increase between R3.2.1 and R3.2.2 affecting mostly the Application software designers role). However in R4 (see Figure 11), we can see a much higher increase in the results of both the CPX and the ADIT metrics (more than double increase between R4.0.1 and R4.1.1) which is mostly related to the introduction of new concepts in R4. This is also the reason for 3-5 times higher increase in the results of the CPX and the ADIT metrics between the releases in R3 and R4.

**Q3: Number of changes:** In order to count the changes needed to be implemented to adopt a new AUTOSAR meta-model release in one project, we compare the results of the NoC, NoCE and the NoCA metrics for all considered releases. Figure 10 shows the total number of changes between each two releases of the AUTOSAR meta-model.

![Number of changes between all releases](image)

We can see that more changes are made between the releases in branch R3.2 and the releases in branch R4.0 / R4.1, than between the releases in branch R3.0 / R3.1 and the releases in branch R4.0 / R4.1. This indicates that the changes done in branch R3.2 are more than just a subset of the changes done in branch R4.0 / R4.1, even though the Maintenance branch R3.2 should be focused only on bug-fixes and back-porting of the most important concepts from the Development branch R4.0 / R4.1. This means that using later releases in one evolution branch requires more changes to be implemented in order to switch to a release in another evolution branch. By calculating the NoC metric for each role separately, we identified that this is particularly expressed for the ECU communication configurators role whereas other roles are less affected. The NoCE and the NoCA metrics (for applicable roles) show similar results.

**Q4: Affected roles:** In order to identify the roles mostly affected by the evolution of the AUTOSAR meta-model, we compare the results of the NoC, NoCE and the NoCA metrics for all considered releases. Figure 12 shows the results of the NoC metric between consecutive releases in R3 and R4 for each role. We can see that the role of the ECU communication configurators is mostly affected by the evolution followed by the role of the Diagnostics configurators (R3.1.1, R3.2.1, R3.2.2 and R4.1.1). The NoCE metric shows similar results while the NoCA metric is not applicable to these two roles as they are mapped to the AR M1 parts of the meta-model.

VI. DISCUSSION AND VALIDATION OF THE RESULTS

In order to validate the conclusions we derived from the measurements, we analyzed the release notes of the considered AUTOSAR releases. The brief summary is shown in Table I.
Table I. Factors Affecting Different Releases

<table>
<thead>
<tr>
<th>Release</th>
<th>Factors</th>
</tr>
</thead>
<tbody>
<tr>
<td>R3.0.1</td>
<td>Bug-fixes, new AR M1 modules (State Mgr), new AR M2 templates (BswModuleTemplate), FIBEX standard harmonization</td>
</tr>
<tr>
<td>R3.0.2 - R3.0.3</td>
<td>Bug-fixes only</td>
</tr>
<tr>
<td>R3.1.1</td>
<td>Bug-fixes, conc. On-Board Diagnostics (AR M1)</td>
</tr>
<tr>
<td>R3.1.2 - R3.1.5</td>
<td>Bug-fixes only</td>
</tr>
<tr>
<td>R3.2.1</td>
<td>Bug-fixes, new concepts Partial networking and Production and development errors, back-ported concepts End2End protection, extended Complex Device Driver, Basic Software Mode Manager and FlexRay ISO Transport protocol modules</td>
</tr>
<tr>
<td>R3.2.2</td>
<td>Bug-fixes only</td>
</tr>
<tr>
<td>R4.0.1</td>
<td>Bug-fixes, new concepts Ethernet, Variant handling, Timing model, etc. meta-model cleanup</td>
</tr>
<tr>
<td>R4.0.2</td>
<td>Bug-fixes, new AR M2 templates (StandardizationTemplate, AutosarTopLevelStructure), new AR M1 module MemMap</td>
</tr>
<tr>
<td>R4.0.3</td>
<td>Bug-fixes, forward-ported concept Partial networking (R3.2.1), new AR M1 module FlexRay AR Transport Protocol, new SPEM UML profile for Methodology</td>
</tr>
<tr>
<td>R4.1.1</td>
<td>Bug-fixes, new concepts Partial networking on Ethernet, continued FIBEX harmonization and Timing model, J1939 for heavy duty vehicles, etc., maintainability improvements (revision of the ECU vs. Local scope of AR M1 parameters)</td>
</tr>
<tr>
<td>R4.1.2</td>
<td>Bug-fixes only</td>
</tr>
</tbody>
</table>

Apart from the new concepts, we believe the meta-model cleanup activity between R3 and R4 is the reason for higher increase in the number of changes needed to be implemented for a switch from a release in R3 to a release in R4.

By analyzing different concepts, we concluded that they mostly affect the ECU communication related parts of the AUTOSAR meta-model (e.g. harmonization with the FIBEX standard used to specify the communication between ECUs, Ethernet as a communication medium, Partial networking for partially switching off the communication). Several concepts are also related to the ECU diagnostics (e.g. On-board diagnostics, Production and development errors). This validates our conclusion that the ECU communication configurators and the Diagnostics configurators roles are mostly affected by the AUTOSAR meta-model evolution and need most re-work.

Even though we designed our case study for analyzing the evolution of the AUTOSAR meta-model, we believe that most of the steps are applicable to a wider range or industrial meta-models based on MOF. Therefore we recommend to other companies who would like to monitor the evolution of meta-models used in their development projects the following:

1) Role based analysis of the meta-model by mapping different roles to the relevant parts of the meta-model.
2) Consideration of the relevant changes only. This is applicable only if the meta-model contains data which does not affect the tools working with the models.
3) Usage of the proposed data-model and the metrics for the analysis of the meta-model evolution. Note that not all data-model parts are applicable to all meta-models, e.g. Generalization Connectors in case of flat meta-model structure.
VII. CONCLUSIONS

In this paper, we present a case study analysis of the AUTOSAR meta-model evolution. The goal of the study is to assist software designers who work with multiple AUTOSAR meta-model releases in planning the adoption of newer releases in the development projects. We achieve this by visualizing the size and the complexity increase between different meta-model releases and calculating the number of changes needed to be implemented in order to adopt a newer release. As these results are based on the quantitative data analysis, they can be fully automated and as such used as an early indicator of possible impact of adopting new meta-model releases on the existing projects and used tooling and also as a preliminary estimate of the effort needed to implement the changes.

In order to understand possible implications of adopting new meta-model releases, we showed the results of the Number of elements, the Number of Attributes, the Complexity and the Average dept of inheritance metrics for each role in each release. For example, a high complexity increase between current and adopting meta-model release for a certain role may have a substantial impact on the quality of the corresponding models instantiating these releases. We showed that the size and the complexity of the AUTOSAR meta-model is increasing between different minor and major releases while it is relatively stable between different revisions. The ECU communication configurators role followed by the Diagnostics configurators role is mostly affected by the changes.

In order to estimate the effort needed to switch from one meta-model release to another, we calculated the Number of changes and the Number of changed elements / The number of changed attributes metrics between each two release of the meta-model. We assume that each change requires a certain implementation effort and therefore more changes between two releases indicate higher effort in switching from one release to another. We showed that the highest effort is needed when switching from a late AUTOSAR meta-model release in one evolution branch to a late release in another evolution branch.

In our future work, we plan to study the evolution of the UML 2.0 meta-model using the same approach as described in this paper. We also plan to assess the applicability of different metrics for measuring the evolution of meta-models.

ACKNOWLEDGMENT

The authors would like to thank Swedish Governmental Agency for Innovation Systems (VINNOVA) for funding the work presented in this paper and the AUTOSAR team at Volvo Car Corporation for contributing to the research.

REFERENCES