Quantifying Long-Term Evolution of Industrial Meta-Models - A Case Study

Darko Durisic
Department of Electrical
Systems Design
Volvo Car Corporation
Gothenburg, Sweden
darko.durisic@volvocars.com

Miroslaw Staron and
Matthias Tichy
Software Engineering Division
Chalmers | University of Gothenburg
Gothenburg, Sweden
firstname.lastname@cse.gu.se

Jörgen Hansson
Software Engineering Division
Chalmers | University of Gothenburg
Gothenburg, Sweden
jorgen.hansson@chalmers.se

***Abstract*—Measurement in software engineering is an important activity for successful planning and management of projects under development. However knowing what to measure and how is crucial for the correct interpretation of the measurement results. In this paper, we assess the applicability of a number of software metrics for measuring a set of meta-model properties - size, length, complexity, coupling and cohesion. The goal is to identify which of these properties are mostly affected by the evolution of industrial meta-models and also which metrics should be used for their successful monitoring. In order to assess the applicability of the chosen set of metrics, we calculate them on a set of releases of the standardized meta-model used in the development of automotive software systems – the AUTOSAR meta-model – in a case study at Volvo Car Corporation. To identify the most applicable metrics, we used Principal Component Analysis (PCA). The results of these metrics shall be used by software designers in planning software development projects based on multiple AUTOSAR meta-model versions. We concluded that the evolution of the AUTOSAR meta-model is quite even with respect to all 5 properties and that the metrics based on fan-in complexity and package cohesion quantify the evolution most accurately.**

I. Introduction

Measuring the properties of software today is an inseparable part of software engineering. As the results of the measurements may have a severe impact on project decisions, choosing the right properties to be measured and the right metrics for their measurement is crucial for the correct interpretation of the measurement results [1]. One particularly important use of software metrics is for monitoring the evolution of software [2]. As meta-models are used to define properties of models and as such they influence the software instantiated from these models [3], monitoring the evolution of the meta-models plays an important role in planning the evolution of the software based on them. The goal of this paper is to identify the most applicable metrics for effective monitoring of the evolution of the industrial meta-models.

Industrial meta-models represent a specific kind of meta-models as they are used to define domain-specific models [4] (e.g. telecommunication, automotive, avionics) which are usually exchanged between a number of stakeholders in the development process. As these stakeholders may use different tools to work with the models, 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 with the same models. For this reason, the evolution of such product oriented 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.

One example of such industrial meta-model is the standardized meta-model used in development of automotive software systems - AUTOSAR (AUTomotive Open System ARchitecture) [5] meta-model. A simplified example of the usage of the AUTOSAR meta-model to allocate software components to Electronic Control Units (ECUs)[1] is shown in Figure 1.

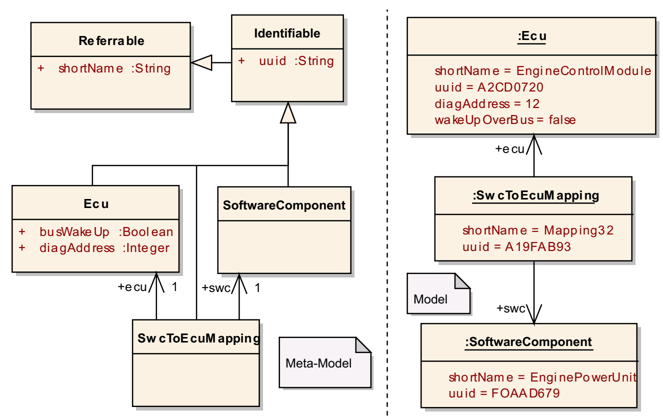

Fig. 1. Example of the AUTOSAR Meta-Model and its usage

In large projects which span over longer period of time (e.g. 4-5 years), monitoring the evolution of meta-models is even more important as multiple versions of one meta-model may need to co-exist in one project [6]. The reason for this is that long life-cycles usually imply the existence of the legacy software based on the old meta-model versions but also the new software based on the new versions. This can be observed in car projects where, due to the distributed nature of the automotive systems, different sub-systems may have their own development cycles so their models may be instantiated from different versions of the AUTOSAR meta-model. Therefore measuring certain properties of meta-models between different versions is important to understand the potential impact of adopting new meta-model versions in terms of compatibility and effort in updating the existing tools and models.

[1] Embedded system (hardware and software) responsible for one or more vehicle functions (e.g. engine control, breaking).

In this paper, we present the assessment of the applicability of a number of software metrics for monitoring 5 properties of meta-model evolution - size, length, complexity, coupling and cohesion [7]. We assess the metrics in a case study of the AUTOSAR meta-model at Volvo Car Corporation. To identify the most applicable metrics, we used Principal Component Analysts (PCA) [8]. The results of these metrics shall be used by software designers for two main purposes: First, to plan the adoption of new AUTOSAR meta-model releases in on-going or future development projects by providing initial estimations about the adoption effort. Second, to predict the impact of adopting new AUTOSAR meta-model releases on the existing models in terms of quality and re-work. Based on the results of the PCA, concluded that the evolution of the AUTOSAR meta-model is quite even with respect to all 5 properties. We also concluded that the metrics based on fan-in complexity and package cohesion quantify the evolution most accurately. This is validated by comparing the results of these metrics to the release notes of each AUTOSAR meta-model release.

The rest of the paper is organized in the following way: Section 2 describes the related work. Section 3 describes the context of the case study - AUTOSAR meta-model. Section 4 describes the design of the case study including the research questions and the research method. Section 5 formally defines the assessed metrics. Section 6 presents the results of the PCA performed on the results of the metrics calculated on a number of releases of the AUTOSAR meta-model. Finally, Section 7 summarizes our conclusions and plans for future work.

II. Related Work

There exist a number of papers today analyzing the evolution of software, especially related to visualization of the software evolution [9]. Some of them focus on the evolution of models, like the one from Madhavi et. al. [10], or they define or analyze the metrics applicable for measuring their properties such as the ones from Hyoseob et. al. [11], Marchesi et. al. [12] and McQuillan et. al. [13]. However not many papers focus on the analysis of the meta-model evolution. Additionally, there is a lack of empirical research in this area, especially related to the evolution of long-term industrial meta-model.

For the definition of metrics, we decided to use formalized definition based on the mathematical model. However there are several other applicable approaches to the formal definition of object-oriented software metrics such as the one proposed by Baroni et. al. using OCL [14], the one proposed by Wakil et. al. using XQuery expressions for XMI documents [15] or the one proposed by Lamrani et. al. using Z language [16].

Finally we use PCA to assess the correlations between different metrics and to identify the metrics which are able to measure the desired properties most accurately. This was the goal of several other papers such as the ones from Del Almo et. al. [17], Dash et. al. [18] and Nagappan et. al. [19].

III. AUTOSAR Meta-Model and its Role

Automotive software systems are distributed systems where one premium vehicle today typically contains around 70 - 100 ECUs [20]. 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 implementation and development, the complexity of the automotive software systems is constantly increasing [21] due to new features in cars [22].

In order to facilitate the distributed development of automotive software systems, the 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 the architectural components in order to standardize the exchange format for their models between different tools. The models are expressed using XML and the XML schema used for the validation of the models is generated from the AUTOSAR meta-model [23] (see the simplified sketch of the AUTOSAR software development process in Figure 2). Therefore the AUTOSAR meta-model is used as a basis for designing different parts of the AUTOSAR architecture.

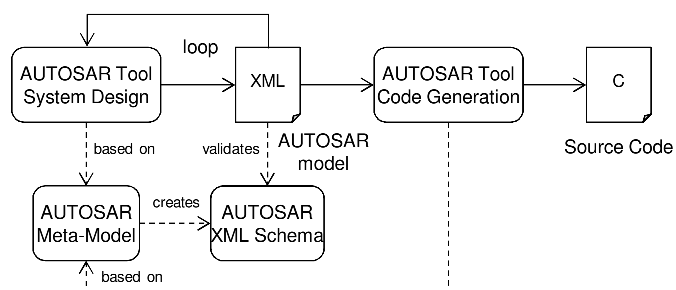

Fig. 2. Automotive software development process based on AUTOSAR

The AUTOSAR meta-model hierarchy is based on the Meta-Object Facility (MOF) standard [24] and it contains 5 meta-layers (4 meta-layers plus MOF). Each meta-layer instantiates the layer above, as depicted in Figure 3.

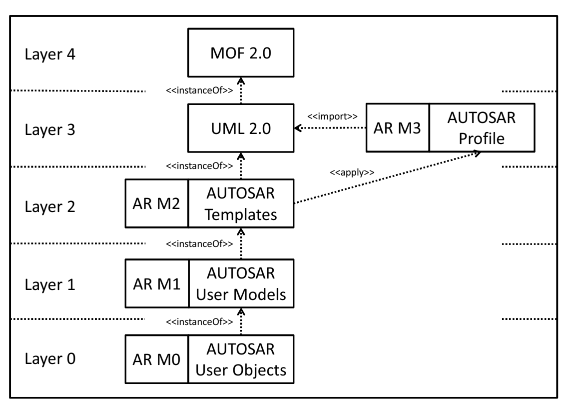

Fig. 3. AUTOSAR meta-model layers

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 *AR M2* meta-model which we refer to as the AUTOSAR meta-model.

The *AR M2* meta-model consists of a hierarchy of classifiers with their attributes and it is divided into a number of top level packages referred to as 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.). Classes in the *AR M2* meta-model may be specialized from multiple classes.

IV. Case Study Design

We conduct a case study analysis [25], [26] of the applicability of a number of software metrics for quantifying the evolution of the AUTOSAR meta-model at Volvo Car Corporation. The formal definition of our research objective is defined according to the structure of Wohlin et. al. [27] as:

- **Goal:** Assess the applicability of a number of metrics for quantifying a set of meta-model properties.
- **Purpose:** Identify the most applicable metrics for monitoring the AUTOSAR meta-model evolution.
- **Field:** Size, length, complexity, coupling and cohesion properties of the meta-model.
- **View:** Software designers working with models instantiating multiple AUTOSAR meta-model versions.
- **Context:** Automotive software systems based on the AUTOSAR standard deployed to Volvo cars.

In order to extract data for the measurements from different AUTOSAR meta-model releases, we defined a meta-data model (simplified version of MOF) presented in Figure 4.

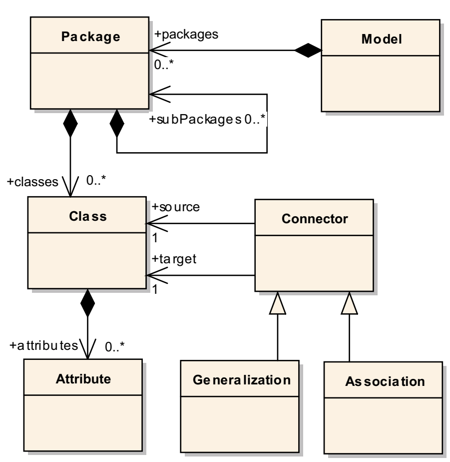

Fig. 4. Meta-data-model used for the measurements

At the top we have a *MetaModel* meta-element which contains a certain number of top level *Package* meta-elements - called templates in the *AR M2*. Templates are used to define how to model one specific part of the automotive electrical system, e.g. *Generic Structure Template* defines generic *Class*es from which all other *Class*es are specialized, the *Software Component Template* defines software components and their interaction, the *System Template* defines communication between ECUs, etc. *Template*s contain a hierarchy of *Package* meta-elements where each *Package* contains *Class* meta-elements and / or other *Package*s. *Class*es contain *Attribute* meta-elements. Finally, binary relations between *Class*es are realized with *Connector* meta-elements which can be either *Generalization*s or *Association*s.

In order to monitor the evolution of the AUTOSAR meta-model, we chose to assess a set of structural object-oriented metrics based on the metrics presented by Genero et. al. [28] and Yi et. al. [29] as they are applicable to Class diagrams which represent building blocks of the AUTOSAR meta-model. We selected 10 metrics presented in Table I. The metrics are categorized according to the 5 properties defined by Briand et. al. [7] - size, length, complexity, coupling and cohesion, and they all satisfy the criteria of the corresponding property. Our goal was to cover each property considering only simple (implementation wise) and easily understandable metrics. Also, the goal was to cover all of the elements of the used meta-data-model presented in Figure 4.

TABLE I. Metrics

Metric	Abbreviation	Property
Number of classes	NOC	Size
Number of attributes	NOA	Size
Depth of inheritance	DIT	Length
Fan-in	FI	Complexity
Fan-out	FO	Complexity
Fan-IO	FIO	Complexity
Package coupling	PCP	Coupling
Coupling between classes	CBC	Coupling
Package cohesion	PCH	Cohesion
Cohesion ration	CR	Cohesion

For the **size** property, we chose the *Number of classes* and the *Number of attributes* metrics. *Class*es represent the main meta-elements of the AUTOSAR *AR M2* meta-model as they give semantics to the objects used in the actual models instantiating the AUTOSAR meta-model, e.g. *ECU*s, *SoftwareComponent*s, *SystemSignal*s, etc. *Attribute*s provide additional information about the *Class*es, e.g. *length* of a *SystemSignal*. As the AUTOSAR meta-model does not contain methods and *Package*s are just logical structures of *Class*es without any semantics, we consider the number of *Class*es and the number of *Attribute*s as the most suitable indicators of the size increase of the AUTOSAR meta-model.

Note that even though in the modeling world *Association*s can be considered as *Attribute*s of the *source Class*es, in case of industrial meta-models they may have slightly different semantics. The reason for this is the fact that *Class*es represent logical entities whose instances may be modeled by separate teams. Therefore the introduction / removal of one *Association* may have globally wider impact than the introduction / removal of one *Attribute* which describes only one logical entity (*Class*). For this reason, we analyzed them in a context of complexity, coupling and cohesion rather than in the context of size. Figure

5 shows an example of the different usage of *Association*s and *Attribute*s in the AUTOSAR meta-model.

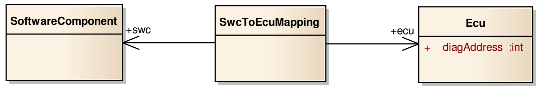

Fig. 5. Different semantics of associations and attributes.

In this example, one *SoftwareComponent* can be allocated onto one *Ecu*. This allocation is captured in another modeling entity *SwcToEcuMapping* which contains *Association*s to both *SoftwareComponent* and *Ecu* entities. Therefore these *Association*s may introduce additional complexity to both *SoftwareComponent* and *Ecu* modeling entities as they may be modeled by separate teams while, for example, the *Attribute diagAddress* describes just one *Ecu* entity (it indicates the ID of the *Ecu* entity used for responses to diagnostic routines) and therefore does not require interaction between different teams.

For the **length** property, we chose the *Depth of inheritance* metric. The reason for this is a deep inheritance hierarchy of *Class*es in the AUTOSAR meta-model where *Class*es at the top are abstract *Class*es used for defining the high level properties of *Class*es below (e.g. *shortName*, *category*, *uuid*, etc). The non-abstract *Class*es may have a hierarchy as well.

For the **complexity** property, we chose the *Fan-in*, *Fan-out* and *Fan-IO* metrics. Generally metrics based on fan-in and fan-out are widely accepted for measuring structural complexity between different modules. Then the fan-in represents the number of modules which are calling a given module while the fan-out represents the number of modules which are called by the given module. As modules in the AUTOSAR meta-model represent *Class*es (or *Packag*es of *Class*es) connected by *Association*s, it is not possible to call one module from another. However since objects of different *Class*es may be part of different domains and as such modeled by separate teams, any interaction between them can be considered as increase in the overall complexity of the models instantiating the AUTOSAR meta-model (see Figure 5 where *SoftwareComponents* modeled by one team can now be allocated onto *Ecu*s modeled by another team). Therefore we consider the *source* of the *Association* as a fan-out property of the referred *Class* and the *target* as a fan-in property of the referred *Class*.

For the **coupling** property, we chose the *Package coupling* and the *Coupling between classes* metrics. Both metrics are based on fan-in and fan-out properties of *Class*es, just *Coupling between classes* metric considers all *Association*s connecting the analyzed *Class* with other *Class*es while *Package coupling* metric considers only the *Association*s connecting the analyzed *Class* with *Class*es from other *Package*s.

Finally for the **cohesion** property, we chose the *Package cohesion* and the *Cohesion ratio* metrics. Both metrics are based on fan-in and fan-out properties of *Class*es explained above, just considering only the *Association*s connecting the analyzed *Class* with *Class*es inside the same *Package*. Please note that *Package cohesion* metric is applicable only to meta-models which are well logically structured into different packages according to their functionality rather than according to other properties such as types vs. prototypes, etc. Imagine the case where we have all data-type *Class*es in one *Package* referred to by *Class*es in other *Package*s. This results in a low cohesion of these *Package*s even though the functional cohesion may be high. As we believe the *AR M2* meta-model is strongly based on the logical wholes starting with the definition of different templates at the top (see the example of the *SystemTemplate* structure in Figure 6), we decided to include this metric in the assessment even though it may not be a good choice for other meta-models.

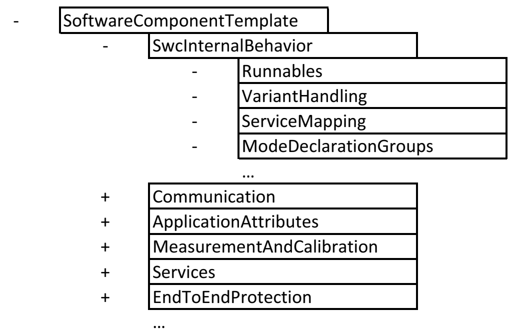

Fig. 6. Example - Software Component Template package structure

In order to assess the applicability of the analyzed software metrics for monitoring the evolution of meta-models, we study their results on the evolution of the AUTOSAR meta-model. We consider a total number of 22 releases of the AUTOSAR meta-model from the very beginning of AUTOSAR until now which represents a period of 8 years. The main goal is to eliminate the metrics with redundant results and also to find the metrics with results which can quantify the evolution of the AUTOSAR meta-model in the most accurate way. In order to achieve this, we performed the PCA to first identify the meaningful principal components and then analyze the importance of the results of each metric in these components. We validated the accuracy of the results of the most important metrics together with the AUTOSAR team from Volvo Cars. We did this by comparing the results of the metrics to their expectations based on the analysis of the release notes for each considered AUTOSAR meta-model release.

We analyzed the releases of the AUTOSAR *M2* meta-model from three different perspectives - the entire *M2* , the *Software Component Template* package of the *M2* and the *System Template* package of the M2. The *Software Component Template* and the *System Template* are the two biggest top level packages of the *M2* meta-model in size. For example, the number of *Class*es of the *Software Component Template* represents on average 31% of the number of *Class*es of the entire *M2* and the number of *Class*es of the *System Template* represents on average 30% of the number of *Class*es of the *M2*. We also included the *Common Structure Template* top level package (on average 11% of the number of *Class*es of the *M2*) in the analysis of both *Software Component Template* and *System Template* packages as its classes are commonly shared between them.

V. Definition of the Metrics

The following sub-sections formally define the chosen metrics based on the meta-data-model presented in Figure 4.

A. Number of classes

In order to define the *Number of classes* metric, we first define the following sets:

- $P(m) = \{p_1(m), p_2(m), ..., p_\alpha(m)\}$ - a set of *Packages* aggregated by *MetaModel* m.
- $P(p) = \{p_1(p), p_2(p), ..., p_\beta(p)\}$ - a set of *Packages* aggregated by *Package* p.
- $C(p) = \{c_1(p), c_2(p), ..., c_\gamma(p)\}$ - a set of *Classes* aggregated by *Package* p.

The *Number of classes* metric for *Package* p is calculated as a sum of (i) the number of classes aggregated by p and (ii) the *Number of classes* of the *Packages* aggregated by p, recursively.

$$NOC(p) = |C(p)| + \sum_{i=1}^{|P(p)|} NOC(p_i(p))$$

The *Number of classes* metric for *MetaModel* m is calculated as the *Number of classes* of the *Packages* aggregated by m.

$$NOC(m) = \sum_{i=1}^{|P(m)|} NOC(p_i(m))$$

B. Number of attributes

In order to define the *Number of attributes* metric, we first define the following additional set:

- $A(c) = \{a_1(c), a_2(c), ..., a_\delta(a)\}$ - a set of *Attributes* aggregated by *Class* c.

The *Number of attributes* metric for *Class* c is calculated as the total number of *Attributes* aggregated by c.

$$NOA(c) = |A(c)|$$

The *Number of attributes* metric for *Package* p is calculated as the sum of (i) the *Number of attributes* of the *Classes* aggregated by p and (ii) the *Number of attributes* of the *Packages* aggregated by p, recursively.

$$NOA(p) = \sum_{i=1}^{|C(p)|} NOA(c_i(p)) + \sum_{i=1}^{|P(p)|} NOA(p_i(p))$$

The *Number of attributes* metric for *MetaModel* m is calculated as the *Number of attributes* of the *Packages* aggregated by m.

$$NOA(m) = \sum_{i=1}^{|P(m)|} NOA(p_i(m))$$

C. Depth of inheritance

In order to define the *Depth of inheritance* metric, we first define the following additional set:

- $C(c) = \{c_1(c), c_2(c), ..., c_\theta(c)\}$ - a set of ('parent') *Classes* connected to *Class* c via *Generalization Connectors*, i.e. *target* of the *Generalization* refers to a *Class* in this set and the *source* refers to c.

The *Depth of inheritance* metric for *Class* c is calculated as the maximum number of *Generalization Connectors* in the inheritance hierarchy starting from the considered *Class* to the ('root') *Classes* with no further parents.

$$DIT(c) = \begin{cases} 0, & C(c) = \emptyset \\ max(\forall c \in C(c) : 1 + DIT(c)), & \text{otherwise} \end{cases}$$

The *Depth of inheritance* metric for *Package* p is calculated as the sum of (i) the *Depth of inheritance* of the *Classes* aggregated by p and (ii) the *Depth of inheritance* of the *Packages* aggregated by p, recursively.

$$DIT(p) = \sum_{i=1}^{|C(p)|} DIT(c_i(p)) + \sum_{i=1}^{|P(p)|} DIT(p_i(p))$$

The *Depth of inheritance* metric for *MetaModel* m is calculated as the *Depth of inheritance* of the *Packages* aggregated by m.

$$DIT(m) = \sum_{i=1}^{|P(m)|} DIT(p_i(m))$$

D. FanIn

In order to define the *Fan-in* metric, we first define the following additional set:

- $SI(c) = \{si_1(c), si_2(c), ..., si_\varepsilon(c)\}$ - a set of *Associations* whose *target* refers to *Class* c. SI is short from 'aSsociation Input'.

The *Fan-in* metric for *Class* c is calculated as the total number of *Associations* whose *target* refers to c.

$$FI(c) = |SI(c)|$$

The *Fan-in* metric for *Package* p is calculated as the sum of (i) the *Fan-in* of the *Classes* aggregated by p and (ii) the *Fan-in* of the *Packages* aggregated by p, recursively.

$$FI(p) = \sum_{i=1}^{|C(p)|} FI(c_i(p)) + \sum_{i=1}^{|P(p)|} FI(p_i(p))$$

The *Fan-in* metric for *MetaModel* m is calculated as the *Fan-in* of the *Packages* aggregated by m.

$$FI(m) = \sum_{i=1}^{|P(m)|} FI(p_i(m))$$

E. FanOut

In order to define the *Fan-out* metric, we first define the following additional set:

- $SO(c) = \{so_1(c), so_2(c), ..., so_\zeta(c)\}$ - a set of *Associations* whose *source* refers to *Class* c. *SO* is short from 'aSsociation Output'.

The *FanOut* metric for *Class* c is calculated as the total number of *Association*s whose *source* refers to c.

$$FO(c) = |SO(c)|$$

The *Fan-out* metric for *Package* p is calculated as the sum of (i) the *Fan-out* of the *Class*es aggregated by p and (ii) the *Fan-out* of the *Package*s aggregated by p, recursively.

$$FO(p) = \sum_{i=1}^{|C(p)|} FO(c_i(p)) + \sum_{i=1}^{|P(p)|} FO(p_i(p))$$

The *Fan-out* metric for *MetaModel* m is calculated as the *Fan-out* of the *Package*s aggregated by m.

$$FO(m) = \sum_{i=1}^{|P(m)|} FO(p_i(m))$$

F. *FanInOut*

The *Fan-IO* metric for one *Class* is calculated as the multiplication of its *FanIn* and *FanOut* values. We chose to multiply *Fan-in* and *Fan-out* inspired by the Henry and Kafura's [30] complexity metric which equals to the squared multiplication of *Fan-in* and *Fan-out*. However we decided to remove the square from the formula due to its unjustified amplification of the results (we explained this more in [21]) and because it does not satisfy the criteria of complexity metrics defined in [7] which we used as basis for defining the metrics. The *Fan-IO* metric for *Class* c is defined as:

$$FIO(c) = FI(c) * FO(c)$$

The *Fan-IO* metric for *Package* p is calculated as the sum of (i) the *Fan-IO* of the *Class*es aggregated by c and (ii) the *Fan-IO* of the *Package*s aggregated by p, recursively.

$$FIO(p) = \sum_{i=1}^{|C(p)|} FIO(c_i(p)) + \sum_{i=1}^{|P(p)|} FIO(p_i(p))$$

The *Fan-IO* metric for *MetaModel* m is calculated as the *Fan-IO* of the *Package*s aggregated by m.

$$FIO(m) = \sum_{i=1}^{|P(m)|} FIO(p_i(m))$$

G. *Package coupling*

In order to define the *Package coupling* metric, we first define the following subsets:

- $SIP(c_x) \subset SI(c_x) \mid \forall s \in SIP(c_x) : s \in SI(c_x) \wedge s \in SO(c_y) \wedge c_x \in C(p_x) \wedge c_y \in C(p_y) \wedge p_x \neq p_y$ - a subset of *Association*s whose *target* refers to *Class* c_x aggregated by *Package* p_x such that their *source* refers to *Class* c_y aggregated by another *Package* p_y. SIP is short from 'aSsociation Input package couPling'.
- $SOP(c_x) \subset SO(c_x) \mid \forall s \in SOP(c_x) : s \in SO(c_x) \wedge s \in SI(c_y) \wedge c_x \in C(p_x) \wedge c_y \in C(p_y) \wedge p_x \neq p_y$ - a subset of *Association*s whose *source* refers to *Class* c_x aggregated by *Package* p_x such that their *target* refers to *Class* c_y aggregated by another *Package* p_y. SOP is short from 'aSsociation Output package couPling'.

The *Package coupling* metric for *Package* p is calculated as the sum of (i) the total number of *Association*s whose *source / target* refers to a *Class* aggregated by p and *target / source* refers to a *Class* aggregated by another *Package*, respectively, and (ii) the *Package coupling* of the *Package*s aggregated by p, recursively.

$$PCP(p) = \sum_{i=1}^{(|C(p)|} (|SIP(c_i(p))| + |SOP(c_i(p))|) + \sum_{i=1}^{|P(p)|} PCP(p_i(p))$$

The *Package coupling* metric for *MetaModel* m is calculated as the *Package coupling* of the *Package*s aggregated by m.

$$PCP(m) = \sum_{i=1}^{|P(m)|} PCP(p_i(m))$$

H. *Coupling between classes*

In order to define the *Coupling between classes* metric, we first define the following additional set:

- $CP(c) = \{cp_1(c), cp_2(c), ..., cp_\eta(c)\}$ - a set of *Class*es where there exists an *Association* whose *source / target* refers to this *Class* and *target / source* refers to c respectively. CP is short from 'Classes couPled'.

The *Coupling between classes* metric for *Class* c is calculated as the total number of *Class*es connected to this class via *Association*s (the *source* of *Association* refers to this *Class* and the *target* refers to c or vice versa).

$$CBC(c) = |CP(c)|$$

The *Coupling between classes* metric for *Package* p is calculated as the sum of (i) the *Coupling between classes* of the *Class*es aggregated by p and (ii) the *Coupling between classes* of the *Package*s aggregated by p, recursively.

$$CBC(p) = \sum_{i=1}^{|C(p)|} CBC(c_i(p)) + \sum_{i=1}^{|P(p)|} CBC(p_i(p))$$

The *Coupling between classes* metric for *MetaModel* m is calculated as the *Coupling between classes* of the *Package*s aggregated by m.

$$CBC(m) = \sum_{i=1}^{|P(m)|} CBC(p_i(m))$$

I. *Package cohesion*

In order to define the *Package cohesion* metric, we first define the following subsets:

- $SIH(c_x) \subset SI(c_x) \mid \forall s \in SIH(c_x) : s \in SI(c_x) \wedge s \in SO(c_y) \wedge c_x \in C(p_x) \wedge c_y \in C(p_x)$

- a subset of *Association*s whose *target* refers to *Class* c_x such that their *source* refers to *Class* c_y which are both aggregated by the same *Package* p_x. SIH is short from 'aSsociation Input package coHesion'.

- $SOH(c_x) \subset SO(c_x) \mid \forall s \in SOH(c_x) : s \in SO(c_x) \wedge s \in SI(c_y) \wedge c_x \in C(p_x) \wedge c_y \in C(p_x)$ - a subset of *Association*s whose *source* refers to *Class* c_x such that their *target* refers to *Class* c_y which are both aggregated by the same *Package* p_x. SOH is short from 'aSsociation Output package coHesion'.

The *Package cohesion* metric for *Package* p is calculated as the sum of (i) the number of *Association*s whose both *source* and *target* refer to a *Class* aggregated by p and (ii) the *Package cohesion* of the *Package*s aggregated by p, recursively.

$$PCH(p) = \sum_{i=1}^{(|C(p)|} |SIH(c_i(p))| + |SOH(c_i(p))|) + \sum_{i=1}^{|P(p)|} PCH(p_i(p))$$

The *Package cohesion* metric for *MetaModel* m is calculated as the *Package cohesion* of the *Package*s aggregated by m.

$$PCH(m) = \sum_{i=1}^{|P(m)|} PCH(p_i(m))$$

J. Cohesion ratio

In order to define the *Cohesion ratio* metric, we first define the following additional subset:

- $CH(c) \subset CP(c) \mid \forall c \in CH(c) : c \in C(p) \wedge c \in C(p)$ - a subset of *Class*es coupled to *Class* c such that they are aggregated by the same *Package* p which aggregates c. CH is short from 'Classes coHered'.

The *Cohesion ratio* metric for *Class* c is calculated as a division of (i) the number of *Class*es connected to c via *Association*s (the *source* of the *Association* refers to this *Class* and the *target* refers to c or vice versa) such that they are aggregated by the same *Package* p which aggregates c and (ii) the number of *Class*es in p.

$$CR(c) = |CH(c)| \; / \; |C(p)| \; ; \; c \in C(p)$$

The *Cohesion ratio* metric for *Package* p is calculated as the sum of (i) the *Cohesion ratio* of the *Class*es aggregated by p and (ii) the *Cohesion ratio* of the *Package*s aggregated by p, recursively.

$$CR(p) = \sum_{i=1}^{|C(p)|} CR(c_i)(p) + \sum_{i=1}^{|P(p)|} CR(p_i(p))$$

The *Cohesion ratio* metric for *MetaModel* m is calculated as the *Cohesion ratio* of the *Package*s aggregated by m.

$$CR(m) = \sum_{i=1}^{|P(m)|} CR(p_i(m))$$

VI. Case Study Results

The following section contains the results of the PCA. As input to the PCA, we used the results of the chosen set of 10 metrics calculated on a set of 22 releases of the AUTOSAR meta-model. We performed 3 different PCA based on the results of the metrics calculated on the releases of the (i) entire *AR M2* meta-model., (ii) *Software Component Template* package of the *M2* only and (iii) *System Template* package of the *M2* only. The results of these 3 PCA are presented in the following sub-sections.

A. PCA of the entire AUTOSAR M2 meta-model

This section presents the results of the Principal Component Analysis (PCA) for which we used as input the results of the chosen set of 10 metrics calculated on a set of 22 releases of the entire *AR M2* meta-model. Figure 7 shows the identified principal components together with the values of their standard deviation, proportion of variance and cumulative proportion of variance. We consider the principal components with the largest proportion of variance as the components which contribute mostly to the results of the metrics, i.e. their significance is the highest.

Principal Components	Standard deviation	Proportion of Variance	Cumulative Proportion
PC1	3,0557	0,9337	0,9337
PC2	0,6711	0,0450	0,9788
PC3	0,3641	0,0133	0,9920
PC4	0,2133	0,0046	0,9966
PC5	0,1507	0,0023	0,9989
PC6	0,0754	0,0006	0,9994
PC7	0,0675	0,0005	0,9999
PC8	0,0356	0,0001	1,0000
PC9	0,0053	0,0000	1,0000
PC10	0,0001	0,0000	1,0000

Fig. 7. Principal components - AUTOSAR M2 meta-model

The proportion of variances of the identified principal components indicates that the first principal component (PC1) contributes with 93.37% to the variation of the results of the calculated metrics while all the other principal components have significantly less influence. Therefore we concluded that only PC1 is meaningful so we continued with the analysis of the importance of the results of each metric in this principal component. As correlation is generally a good sign of redundancy, we started by investigating the correlation between the results of each two pairs of metrics. Figure 8 shows both the importance of the results of each metric in the the PC1 (table to the left) and the correlation between each two pairs of metrics (table to the right).

	PC1
NOA	0,2725
NOC	0,3194
PCH	0,3262
FI	0,3261
PCP	0,3217
CR	0,3160
DIT	0,3187
FIO	0,3226
CBO	0,3209
FO	0,3147

	NOA	NOC	PCH	FI	PCP	CR	DIT	FIO	CBO	FO
NOA	1,0000	0,9191	0,8292	0,8283	0,8090	0,7645	0,7688	0,7490	0,7343	0,6798
NOC	0,9191	1,0000	0,9757	0,9753	0,9646	0,9294	0,9234	0,9315	0,9254	0,8904
PCH	0,8292	0,9757	1,0000	1,0000	0,9914	0,9515	0,9566	0,9764	0,9811	0,9575
FI	0,8283	0,9753	1,0000	1,0000	0,9916	0,9512	0,9564	0,9764	0,9814	0,9578
PCP	0,8090	0,9646	0,9914	0,9916	1,0000	0,9202	0,9298	0,9628	0,9720	0,9457
CR	0,7645	0,9294	0,9515	0,9512	0,9202	1,0000	0,9736	0,9726	0,9327	0,9206
DIT	0,7688	0,9234	0,9566	0,9564	0,9298	0,9736	1,0000	0,9868	0,9525	0,9471
FIO	0,7490	0,9315	0,9764	0,9764	0,9628	0,9726	0,9868	1,0000	0,9779	0,9706
CBO	0,7343	0,9254	0,9811	0,9814	0,9720	0,9327	0,9525	0,9779	1,0000	0,9945
FO	0,6798	0,8904	0,9575	0,9578	0,9457	0,9206	0,9471	0,9706	0,9945	1,0000

Fig. 8. Metrics correlation - M2 meta-model

By analyzing these results, we concluded that the evolution of the AUTOSAR *M2* meta-model is quite even with respect to all five considered properties. We came to this conclusion based on the high correlation between the *Number of classes* (size), *Depth of inheritance* (length), *Fan-in* / *Fan-out* / *FanIO* (complexity), *Package coupling* / *Coupling between objects* (coupling) and the *Package cohesion* / *Cohesion ratio* (cohesion) metrics.

We also concluded that for quantifying the evolution of the *AR M2* meta-model, it is enough to use only one metric, preferably either the *Package cohesion* or the *Fan-in*. We came to this conclusion based on the high correlation between the results of all metrics except for the results of the *Number of attributes* metric which has lower significance. The choice of the *Package cohesion* or the *Fan-in* metric is based on the highest significance of their results.

B. Software Component Template

This section presents the results of the PCA for which we used as input the results of the chosen set of 10 metrics calculated on a set of 22 releases of the *Software Component Template* package of the *AR M2* meta-model. The proportion of variances of the identified principal components is very similar to the results of the PCA for the entire *AR M2* meta-model. This means that we again identified only one meaningful principal component (PC1) which this time contributes with 96.84% to the variation of the results of the calculated metrics. Figure 9 shows both the importance of the results of each metric in the PC1 (table to the left) and the correlation between each two pairs of metrics (table to the right).

	PC1
NOA	0,1988
FOUT	0,3159
NOC	0,3282
CBO	0,3375
PCP	0,3413
FIN	0,3429
PCH	0,3429
FIO	0,3315
DIT	0,3050
CR	0,2906

	NOA	FOUT	NOC	CBO	PCP	FIN	PCH	FIO	DIT	CR
NOA	1,0000	0,7269	0,7181	0,6440	0,3869	0,3825	0,3825	0,3605	0,3852	0,3154
FOUT	0,7269	1,0000	0,9321	0,9801	0,8328	0,8174	0,8174	0,7401	0,6555	0,5578
NOC	0,7181	0,9321	1,0000	0,9575	0,8775	0,8783	0,8783	0,7686	0,6692	0,6846
CBO	0,6440	0,9801	0,9575	1,0000	0,9237	0,9135	0,9135	0,8309	0,7321	0,6677
PCP	0,3869	0,8328	0,8775	0,9237	1,0000	0,9957	0,9957	0,9316	0,8276	0,7869
FIN	0,3825	0,8174	0,8783	0,9135	0,9957	1,0000	1,0000	0,9426	0,8432	0,8305
PCH	0,3825	0,8174	0,8783	0,9135	0,9957	1,0000	1,0000	0,9426	0,8432	0,8305
FIO	0,3605	0,7401	0,7686	0,8309	0,9316	0,9426	0,9426	1,0000	0,9501	0,8637
DIT	0,3852	0,6555	0,6692	0,7321	0,8276	0,8432	0,8432	0,9501	1,0000	0,7912
CR	0,3154	0,5578	0,6846	0,6677	0,7869	0,8305	0,8305	0,8637	0,7912	1,0000

Fig. 9. Metrics correlation - Software Component Template

By analyzing these results, we came to the same conclusions as when analyzing the PCA results for the entire *AR M2* meta-model - relatively even evolution of the *Software Component Template* package with respect to all five considered properties where only one metric (*Package cohesion* or *Fan-in*) is enough for its successful quantification. In addition to this, we identified that the *Cohesion ratio* metric, apart from the *Number of attributes*, is also less correlated with the results of other metrics and has lower significance in the PC1.

C. System template

This section presents the results of the PCA for which we used as input the results of the chosen set of 10 metrics calculated on a set of 22 releases of the *System Template* package of the *AR M2* meta-model. The proportion of variances of the identified principal components is very similar to the results of the PCA for the entire *AR M2* meta-model and the *Software Component Template* package. This means that we again identified only one meaningful principal component (PC1) which this time contributes with 97.00% to the variation of the results of the calculated metrics. Figure 10 shows both the importance of the results of each metric in the PC1 (table to the left) and the correlation between each two pairs of metrics (table to the right).

	PC1
PCP	0,2799
FIN	0,3282
PCH	0,3282
CBO	0,3289
FIO	0,3144
FOUT	0,3281
NOC	0,3266
NOA	0,3173
DIT	0,3165
CR	0,2899

	PCP	FIN	PCH	CBO	FIO	FOUT	NOC	NOA	DIT	CR
PCP	1,0000	0,9294	0,9294	0,8566	0,7503	0,7910	0,7580	0,6929	0,6453	0,4665
FIN	0,9294	1,0000	1,0000	0,9737	0,9003	0,9456	0,9270	0,8862	0,8553	0,7438
PCH	0,9294	1,0000	1,0000	0,9737	0,9003	0,9456	0,9270	0,8862	0,8553	0,7438
CBO	0,8566	0,9737	0,9737	1,0000	0,9693	0,9925	0,9030	0,8594	0,8722	0,7753
FIO	0,7503	0,9003	0,9003	0,9693	1,0000	0,9822	0,8376	0,7968	0,8708	0,7588
FOUT	0,7910	0,9456	0,9456	0,9925	0,9822	1,0000	0,9062	0,8676	0,9009	0,8194
NOC	0,7580	0,9270	0,9270	0,9030	0,8376	0,9062	1,0000	0,9864	0,9557	0,9174
NOA	0,6929	0,8862	0,8862	0,8594	0,7968	0,8676	0,9864	1,0000	0,9467	0,9371
DIT	0,6453	0,8553	0,8553	0,8722	0,8708	0,9009	0,9557	0,9467	1,0000	0,9298
CR	0,4665	0,7438	0,7438	0,7753	0,7588	0,8194	0,9174	0,9371	0,9298	1,0000

Fig. 10. Metrics correlation - System Template

By analyzing these results, we came to the same conclusions as when analyzing the PCA results for the entire *AR M2* meta-model and the *Software Component Template* package - relatively even evolution of the *System Template* package with respect to all five considered properties where only one metric (*Package cohesion* or *Fan-in*) is enough for its successful quantification. In addition to this, we identified that the *Cohesion ratio* metric, together with the *Number of attributes*, is not well correlated with the results of other metrics and has lower significance in the PC1.

D. Summary and validation of the metrics results

By analyzing the results of the PCA for the metrics calculated on the entire *AR M2* meta-model and its two biggest packages, *Software Component Template* and the *System Template*, we observed that they are very similar. This is expected as they are based on the same design principles (e.g. logical structuring of *Class*es into *Package*s, low coupling between *Package*s, etc.). Therefore we concluded the following:

1) The evolution of the AUTOSAR meta-model is quite even with respect to all 5 analyzed properties (size, length, complexity, coupling and cohesion).
2) The correlation between the results of the *Number of classes*, *Depth of inheritance*, *Fan-in*, *Fan-out*, *FanIO*, *Package coupling*, *Coupling between objects* and the *Package cohesion* metrics is high while the results of the *Number of attributes* and the *Cohesion ratio* (in case of the *Software Component Template* and the *System Template* packages) metrics are not very correlated to the results of the other metrics.
3) The results of the *Fan-in* and the *Package cohesion* metrics are the most significant for monitoring the evolution of the AUTOSAR meta-model while the results of the *Number of attributes* and the *Cohesion ratio* (in case of the *Software Component Template* and the *System Template* packages of the AUTOSAR *M2* meta-model) metrics are the least significant.

These conclusions can be explained by the strict design principles of AUTOSAR. Namely, *Class*es represent the main modeling units of semantics in the AUTOSAR meta-model and the goal is to keep their complexity, coupling and cohesion as low as possible. That is why *Class*es usually do not have many *Association*s. This assures high correlation between their growth in size and complexity, cohesion and coupling as there are not many highly coupled areas with only a few *Class*es and vice versa. The correlation between the growth in size and the growth in length of *Class*es is implied by the existence of

TABLE II. SUMMARY OF THE RELEASE NOTES

Release	Factors
R1.0	First release
R2.0	Bug-fixes only
R2.1	Bug-fixes, new features in the *Software Component Template* and the *System Template* packages, e.g. *Measurement and calibration*
R3.0.1	Meta-model cleanup, bug-fixes, new template *BswModuleTemplate*, *FIBEX* standard harmonization
R3.0.2 - R3.1.5	Bug-fixes, new concept *On-Board Diagnostics* in R3.1.1 (affected mostly the *AR M1* meta-model, not the analyzed *AR M2*)
R3.2.1	Bug-fixes, new concepts *Partial networking*, *Production and development errors*, *End2End protection*, extended *Complex Device Driver*
R3.2.2	Bug-fixes only
R4.0.1	Meta-model cleanup, bug-fixes, many new concepts such as *Ethernet*, *Variant handling*, *Timing model*, etc.
R4.0.2	Bug-fixes, new *AR M2* templates *StandardizationTemplate* and *AutosarTopLevelStructure*
R4.0.3	Bug-fixes, new concept *Partial networking*
R4.1.1	Bug-fixes, many new concepts such as *Partial networking on Ethernet*, continued *FIBEX harmonization* and *Timing model*, *J1939* for heavy duty vehicles, etc.
R4.1.2	Bug-fixes only

a well established hierarchy of *Class*es (e.g. *Referrable* and *Identifiable Class*es in the example in Figure 1) so each newly introduced *Class* is already a child of several other *Class*es.

The difference in the results of the *Number of attributes* metric in comparison to other metrics can be explained with the fact that *Class*es, as the main modeling units of semantics in the AUTOSAR meta-model, may or may not contain additional descriptions in the form of *Attribute*s (there are many *Class*es without *Attribute*s, e.g. *SwcToEcuMapping* from Figure 1). This depends on the logic of *Class*es, not their number, so the high increase in the *Number of classes* does not necessarily mean the high increase in the *Number of attributes*.

In order to validate the accuracy of the *Fan-in* and the *Package cohesion* metrics, we studied the release notes of the considered AUTOSAR meta-model releases in order to compare them to the results of these two metrics. A brief summary of the release notes is shown in Table II and the trend in the results of the *Fan-in* metric calculated on the *AR M2* meta-model releases is shown in Figure 11. The trend in the results of the *Package cohesion* metric is very similar due to high correlation between the results of these two metrics.

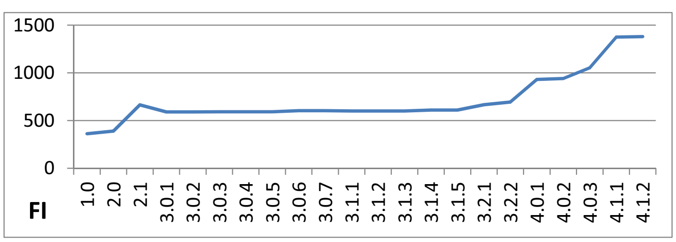

Fig. 11. Fan-in trend - M2 meta-model

We can see relatively stable results of the *Fan-in* metric in releases 3.0.2 - 3.1.5 and also between releases 3.2.1 and 3.2.2 and between releases 4.1.1 and 4.1.2. This is expected as these releases contain bug-fixes only, i.e. no new features are introduced. On the other hand we can see an increase in the *Fan-in* metric results between releases 2.0 and 2.1, 4.0.2 and 4.0.3 and between releases 4.0.3 and 4.1.1 due to the introduction of new concepts. This is also expected as concepts are used to incorporate new features into the AUTOSAR meta-model. Finally we concluded that a decrease in the results of the *Fan-in* metric between releases 2.1 and 3.0.1 is related to the meta-model cleanup activity which removed the unused / obsolete elements from the meta-model. Similar cleanup activity occurred in release 4.0.1 but due to many new concepts incorporated into this release, the results of the *Fan-in* metric are still increased.

E. Recommendations

Due to the logical organization of the AUTOSAR meta-model structured into different packages where each package may be used to define the properties of the models developed by separate teams, monitoring the evolution of the AUTOSAR meta-model shall be done per package bases. We present here the analysis of the top level packages of the *AR M2* - *Software Component Template* and *System Template* - but similar analysis can be done on the packages situated lower in the hierarchy. Therefore, we recommend to the software designers of one team who plan to adopt a newer release of the AUTOSAR meta-model to analyze the changes in the relevant packages between the current and the new release according to the following steps:

1) Measure the complexity growth using the *Fan-in* metric. These results shall be used to indicate the complexity increase of the software models and tools working with the models after the adoption of the new AUTOSAR meta-model release.
2) Together with measuring the complexity, we propose to measure the increase in the *Package cohesion* of the relevant packages in order to estimate the work needed to be done internally in one team.
3) In order to estimate possible integration issues between different teams and their tools, we propose to measure the increase in the *Package coupling* of the relevant packages as it shows the growth in communication between different packages which may be developed by separate teams.
4) Finally we propose to complement the results of these metrics (*Fan-in*, *Package cohesion* and *Package coupling*) by measuring the size increase of the relevant packages using the *Number of classes* metric. The reason for this is to assure that the metrics are in proportion (as the PCA show) as otherwise disruptive changes may have occurred in the meta-model which may require dedicated task force to implement.

Despite the fact that we defined and analyzed the results of the assessed metrics in a case study of AUTOSAR meta-model, we believe they are applicable for quantifying the evolution of a larger set of meta-models based on MOF, e.g. the UML meta-model. This is especially the case with the domain specific meta-models which are used to define the models exchanged between different parties in the development process where the distinction between the cohesion (e.g. attributes and connectors connecting the classes inside one package) and coupling properties (e.g. connectors connecting the classes in different packages) is very important. However depending on the logical structure of the analyzed meta-model, different metrics may have different significance in quantifying the meta-model evolution and also not all meta-model properties (e.g. size and complexity) may be equally affected.

VII. Conclusion

In this paper, we assessed the applicability of 10 different metrics for quantifying the evolution of meta-models with respect to 5 properties - size, length, complexity coupling and cohesion. We assessed the metrics in a case of AUTOSAR meta-model evolution at Volvo Car Corporation. The goal was to identify which of these properties are mostly affected by the evolution of the AUTOSAR meta-model and which of the assessed metrics are able to monitor them most accurately. In order to do this, we performed the Principal Component Analysis (PCA) of the results of the metrics calculated on a set of 22 releases of the AUTOSAR meta-model. We validated the chosen metrics by comparing their results with the release notes of the considered AUTOSAR meta-model releases.

We concluded that the *Fan-in* and the *Package cohesion* metrics provide the most accurate results and that the *Number of attributes* and the *Cohesion ratio* metrics provide the least accurate results. We also concluded that the majority of the metrics, except for the *Number of attributes* and the *Cohesion ratio*, are very correlated which indicates that the evolution of the AUTOSAR meta-model is quite even for all 5 analyzed properties. Based on this, we concluded that it is enough to use only one metric for quantifying the evolution of the AUTOSAR meta-model. Due to the highest accuracy of their results, we propose to use either the *Fan-in* or the *Package cohesion* metric. Finally, we made recommendations on how to combine the results of the assessed metrics to analyze the potential impact of adopting new AUTOSAR meta-model releases.

In our future work, we plan to use the metrics described in this paper to analyze the evolution of the UML 2.0 meta-model. We also plan to develop a method for estimating the effort needed to adopt a newer AUTOSAR meta-model release based on the results of the proposed metrics.

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

[1] R. D. Austin, *Measuring and Managing Performance in Organizations*. Dorset House Publishing, 1996.

[2] M. Lanza and S. Ducasse, "Understanding Software Evolution Using a Combination of Software Visualization and Software Metrics," in *In Proceedings of Langages et Modéles à Objets Conference*, 2002, pp. 135–149.

[3] T. Kűhne, "Matters of (Meta-) Modeling," *Journal of Software and Systems Modeling*, vol. 5, no. 4, pp. 369–385, 2006.

[4] G. Nordstrom, B. Dawant, D. M. Wilkes, and G. Karsai, "Metamodeling - Rapid Design and Evolution of Domain-Specific Modeling Environments," in *Proceedings of the IEEE Conference on Engineering of Computer Based Systems*, 1999, pp. 68–74.

[5] *AUTOSAR Standard*, www.autosar.org, 2003.

[6] S. Becker, B. Gruschko, T. Goldschmidt, and H. Koziolek, "A Process Model and Classification Scheme for Semi-Automatic Meta-Model Evolution," in *MDD, SOA und IT-Management Workshop*, 2007, pp. 35–46.

[7] L. Briand, S. Morasca, and V. Basili, "Property-based Software Engineering Measurement," *IEEE Transactions on Software Engineering*, vol. 22, no. 1, pp. 68–86, 1996.

[8] J. Jolliffe, *Principal Component Analysis*. John Wiley & Sons, 2005.

[9] R. L. Novais, A. Torres, T. S. Mendes, M. Mendonça, and N. Zazworka, "Software Evolution Visualization: A Systematic Mapping Study," *Information and Software Technology*, vol. 55, no. 11, pp. 1860–1883, 2013.

[10] K. Madhavi, "A Framework for Visualizing Model-Driven Software Evolution," in *Advance Computing Conference*, 2009, pp. 1628–1633.

[11] K. Hyoseob and C. Boldyreff, "Developing Software Metrics Applicable to UML Models," in *Proceedings of the 6th Workshop on Quantitative Approaches in Object-Oriented Software Engineering*, 2002.

[12] M. Marchesi, "OOA Metrics for the Unified Modeling Language," in *Proceedings of the Second Euromicro Conference on Software Maintenance and Reengineering*, 1998, pp. 67–73.

[13] J. A. McQuillan and J. F. Power, "On the Application of Software Metrics to UML Models," in *Proceedings of the 2006 international conference on Models in Software Engineering*, 2007, pp. 217–226.

[14] A. Baroni, S. Braz, and F. Abreu, "Using OCL to Formalize Object-Oriented Design Metrics Definitions," in *In Workshop on Quantitative Approaches in Object-Oriented Software Engineering*, 2002.

[15] M. E. Wakil, A. E. Bastawissi, M. Boshra, and A. Fahmy, "A novel approach to formalize and collect Object-Oriented Design-Metrics," in *In Proceedings of the 9th International Conference on Empirical Assessment in Software Engineering*, 2005.

[16] M. Lamrani, Y. Amrani, and Y. Ettouhami, "Formal Specification of Software Design Metrics," in *In Proceedings of the 6th International Conference on Software Engineering Advances*, 2011, pp. 348–355.

[17] C. D. Alamo, D. Pizarro, and R. Pinto, "Discovery of Patterns in Software Metrics using Clustering Techniques," in *In Proceedings of the 38th Conferencia Latinoamericana En Informatica*, 2012, pp. 1–7.

[18] Y. Dash and S. Dubey, "Application of Principal Component Analysis in Software Quality Improvement," *Advanced Research in Computer Science and Software Engineering*, vol. 2, no. 4, pp. 202–205, 2012.

[19] N. Nagappan, T. Ball, and A. Zeller, "Mining Metrics to Predict Component Failures," in *In Proceedings of the 28th International Conference on Software Engineering*, 2006, p. 119125.

[20] M. Broy, "Challenges in Automotive Software Engineering," in *Proceedings of the 28th international conference on Software engineering*, 2006, pp. 33–42.

[21] D. Durisic, M. Nilsson, M. Staron, and J. Hansson, "Measuring the Impact of Changes to the Complexity and Coupling Properties of Automotive Software Systems," *Journal of Systems and Software*, vol. 86, no. 5, pp. 275–1293, 2013.

[22] M. Broy, I. Kruger, A. Pretschner, and C. Salzmann, "Engineering Automotive Software," in *Proceedings of the IEEE*, ser. 2, vol. 95, 2007.

[23] C. David, *Modelling XML Applications with UML Practical e-Business Applications*. Addison-Wesley Professional; 1 edition, 2001.

[24] *OMG. MOF 2.0 Core Final Adopted Specification*, Object Management Group, www.omg.org, 2004.

[25] B. Kitchenham, S. Pfleeger, L. Pickard, L. Jones, P. Hoaglin, K. Emam, and J. Rosenberg, "Preliminary Guidelines for Empirical Research in Software Engineering," *Journal of IEEE Transactions on Software Engineering*, vol. 28, no. 8, pp. 721–734, 2002.

[26] P. Runeson, M. Hőst, A. Rainer, and B. Regnell, *Case Study Research in Software Engineering: Guidelines and Examples*. John Wiley & Sons, 2012.

[27] C. Wohlin, P. Runeson, M. Hőst, M. Ohlsson, B. Regnell, and A. Wesslen, *Experimentation in Software Engineering*. Springer Heidelberg, 2012.

[28] M. Genero, M. Piattini, and C. Calero, "Early Measures for UML Class Diagrams," in *L'OBJET: Software, Databases, Networks*, ser. 4, vol. 6, 2000.

[29] T. Yi, F. Wu, and C. Gan, "A Comparison of Metrics for UML Class Diagrams," in *ACM SIGSOFT Software Engineering Notes*, ser. 1-6, vol. 29, 2004.

[30] S. Henry and D. Kafura, "Software Structure Metrics Based on Information Flow," *Journal of IEEE Transactions on Software Engineering*, vol. 7, no. 5, pp. 510–518, 1981.