Identifying Optimal Sets of Standardized Architectural Features - A Method and its Automotive Application

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

Miroslaw Staron
Software Engineering Division
Chalmers | University of Gothenburg
Gothenburg, Sweden
Miroslaw.Staron@cse.gu.se

Matthias Tichy
Software Engineering Division
Chalmers | University of Gothenburg
Gothenburg, Sweden
Matthias.Tichy@cse.gu.se

ABSTRACT

Industrial standards are used to formalize procedures, rules and guidelines for the industry to follow. Following a standard requires continuous adoption of the new standardized features where only their subset is required by individual companies. Therefore the prioritization of the features and the assessment of their impact on the development projects is crucial for the success of the project. In software engineering, industrial standards are used increasingly often to standardize a language for designing architectural components of the system by defining domain-specific meta-models. The purpose is to assure the interoperability between a number of software tools exchanging the architectural models. In this paper, we present a method for identifying optimal sets of new standardized architectural features to be adopted in the development projects. The optimization is done based on the assessment of their benefit for the projects and the estimated cost of re-work in the modeling tools according to the changes in the standardized meta-model. We evaluate the method by applying it on 14 new architectural features of a new release of the AUTOSAR standard which is followed in the development of the automotive software systems.

Categories and Subject Descriptors

D.2.11 [**Software Engineering**]: Software Architectures—*Domain-specific architectures*; D.2.9 [**Software Engineering**]: Management—*Software Configuration Management*

General Terms

Management, Standardization, Measurement

Keywords

Architectural features, industrial meta-models, change management, optimization

1. INTRODUCTION

Industrial standards are often followed in the design and implementation of architectural components of large software systems [1]. There are two reasons for such a trend. First, the system becomes more reliable due to the use of common architectural components verified in different products. Second, the development cost and time is reduced due to increased re-usability of the architectural components and their implementations. These two reasons are especially important for distributed embedded systems - reliability due to constant increase in the size and complexity of the architectural components (see examples in the automotive domain [2, 3]) and re-usability in order to reduce high development cost related to the hardware and middleware [4].

Apart from their distributed implementation, the development of large distributed embedded systems such as automotive systems is often distributed as well involving a number of actors in the development process. On the one side we have OEMs (Original Equipment Manufacturers) responsible for designing and verifying the architectural components of the system. On the other side we have a chain of suppliers (e.g. application, middleware, hardware, tool suppliers) responsible for their implementation. These actors communicate by exchanging the architectural models and they may use a number of different software tools to work with them. In order to assure the interoperability between these tools, domain-specific meta-models are defined and standardized requiring a full compliance of the architectural models to their meta-models.

The design of the architectural components based on the standardized meta-model requires the standardization of the architectural features before their utilization in the development projects. For this reason, the OEMs and their suppliers constantly incorporate new features into the standard causing thousands of changes to the standardized meta-model. This makes it hardly feasible for the OEMs to adopt all new features from one release of the standard in the development projects. For this reason, the OEMs are usually required to make a prioritization of the architectural features in order to select their optimal subset.

The obvious question that arises is which set of features shall be adopted, i.e. how to balance the need for the new architectural features with the cost of their support in the internal and external (i.e. supplier) tools? Some features are well planned in advance and sometimes even driven by the OEM in the process of their standardization. These features are usually selected for implementation. Some features are not applicable or required by the projects and therefore rejected. However there may be other features which could have a positive impact on the final product but are not required for achieving the predefined goals. For these architectural features, a tradeoff analysis with respect to their cost and benefit shall be performed.

One of the most important factors used in the tradeoff analysis is the impact of the new features on the tools used for working with the architectural models. The main reason for this is the cost of updating the tools to be able to work with the new features but also possible interoperability issues between different tools which may be caused by the changes. As these tools are based on the standardized meta-model, analyzing the evolution of the meta-model with respect to the changes imposed by the new features could be a good indicator of the potential impact of the features on the tools. However due to possibly large number of meta-model changes combined with the common time and budget limitations, tools support is needed for such analysis.

In order to support the cost-benefit analysis of adopting new features in the development projects, we define the following two research questions:

- **Q1:** How to assess the impact of different architectural features on the used domain-specific meta-models?
- **Q2:** How to select the optimum set of features to be adopted based on the assessed impact?

In order to provide answers to the posed research questions, we define a method - *MeFiA* (Meta-model Feature Impact Assessment method) - for quantifying the evolution of industrial meta-models related to a specified set of architectural features. The goal of the method is to identify the optimal sets of new architectural features to be adopted in the development projects based on the assessment of their impact on the standardized meta-models used for modeling the architectural components of the system. We assess the method by applying it on a set of 14 new features of a new release of the AUTOSAR (AUTomotive Open System ARchitecture) standard [5] which is followed in the design of the automotive software architectures. To automate the entire process, we developed a tool which is used at Volvo Cars.

The rest of the paper is structured as follows: Section 2 describes the related work. Section 3 describes the research methodology and research questions. Section 4 defines the *MeFiA* method. Section 5 describes the automotive domain used for the evaluation of the proposed method. Section 6 shows the results of applying the *MeFiA* method on on the automotive domain. Finally, Section 7 summarizes our conclusions and describes our plans for future work.

2. RELATED WORK

The proposed *MeFiA* method shares some similarities with the ATAM (Architecture Tradeoff Analysis Method) [6] based on the CBAM (Cost-Benefit Analysis Method) [7], in particular gathering stakeholders (system architects) and discussing tradeoffs between different architectural solutions affecting the system. However we base our tradeoff analysis on the cost-benefit analysis of the adoption of different sets of architectural features rather than on the design of different architectural solutions.

With respect to the optimized selection of features in different product lines, Asadi et al. [8] propose a framework for automated selection of feature sets based on the functional and non-functional requirements of the system. Related to the architectural features, White et al. [9] present a method for selecting highly optimal sets of architectural features based on their resource consumption. However we are not aware of any work related to the search for optimal set of architectural features to be adopted in the development projects based on their impact on the domain-specific meta-models and software modeling tools based on them.

This papers also contributes to the area of change management of software artifacts developed in distributed working groups. The development of these artifacts (e.g. models, specifications, code) usually relies on the existence of a change management tool containing the requests for changes and a database with the historical versions of the artifacts. Considering the links between these two tools, Bachmann et. al [10] discuss the problems of unreported bugs and missing links in the software repository commits and propose a tool - Linkster - to automatically recover the missing links. Fischer et. al [11] analyze these links in order to find dependencies between features. Our focus is on the utilization of these links to predicting the effort needed to adopt new architectural features in the development projects.

With respect to the impact of the software architecture evolution on the development projects, Gustavsson et al. [12] present the automotive study of how system architects manage architectural changes in different product lines. Eklund et al. [13] discuss the architectural concerns of extending the existing software system with new features. In the automotive domain, Dersten et al. [14] present a systematic literature review of the effects of re-factoring the AUTOSAR architecture such as lower complexity and increased efficiency. Soubra et al. [15] use functional size measurement to estimate the development effort for Electronic Control Units (ECUs)[1] based on the AUTOSAR architecture. Our paper shall be considered as complementary to these studies.

3. RESEARCH METHODOLOGY

We define our research objective according to the structure presented by Wohlin et. al. [16] as:

- **Goal:** Identifying the optimal sets of standardized architectural features to be adopted in the projects.
- **Purpose:** Facilitate the decision making process of which features shall be selected.
- **View:** Project managers and system architects working on software development projects.
- **Context:** The evolution of the standardized meta-model used in the design of architectural components.

This objective is tightly related to the industrial need of car manufacturers working with the architectural models based on the AUTOSAR standard as they often have to make trade-offs between adopting new standardized architectural features and the cost of their implementation. Therefore a method and a tool for the automated feature impact assessment on the modeling tools used in the development and presenting a set of optimal solutions can be very helpful in the decision making process of which new features

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

to select. As the problems identified in the automotive domain served as a motivation for this research, we define our research questions focusing on the development of the automotive software systems. However we believe that other domains such as avionics may face similar problems.

- **Q1:** How to assess the impact of different architectural features on the tools working with the models of the automotive architectural components?
- **Q2:** How to select the optimal set of features to be implemented in the automotive development projects considering their impact on different tools?

In order to provide answers to the posed research questions, we define a method - *MeFiA* - for identifying the optimal sets of architectural features to be adopted in the development projects. The method is developed in 3 steps:

A. Define how to link AUTOSAR meta-model changes to different AUTOSAR features.

   We conduct a case study analysis [17] of the AUTOSAR development process in order to identify means of linking the AUTOSAR meta-model changes to the AUTOSAR features. We define an approach based on the links between a change management tool (*Bugzilla*) containing the description of the features and a software code/model repository (*SVN*) containing different meta-model versions.

B. Define a measure of impact of an AUTOSAR feature on the AUTOSAR based tools.

   As AUTOSAR based tools are based on the AUTOSAR meta-model, quantitative analysis [18] of the changes in the AUTOSAR meta-model related to a particular feature could indicate potential re-work required in the tools to support this feature. For measuring the amount of changes between different releases of the AUTOSAR meta-model, we use the *Number of changes* (*NoC*) metric which has shown to be an effective measure of the size change [19]. We consider only the changes which require certain implementation/integration effort in the AUTOSAR based tools. This is in contrast to the editorial changes (e.g. in the notes of the elements) or changes in the auxiliary parts of the AUTOSAR meta-model (e.g. *Methodology*).

C. Define how to identify the optimal sets of features to be adopted in the development projects.

   For identifying the optimal sets of features to be adopted in the development projects based on their cost-benefit analysis, we follow the approach based on Pareto optimality. When searching for the optimal sets of features, we consider their impact on the entire AUTOSAR meta-model but also on its separate parts related to the most important roles in the automotive software development process. We used the roles and the mapping of the roles to different parts of the AUTOSAR meta-model presented in [19].

We assess the proposed method by applying it on an automotive scenario where the optimal set of features from the AUTOSAR release *4.2.1* shall be identified. In order to extract the relevant data from the AUTOSAR meta-model, perform feature related calculations and present the optimal solutions, we developed a tool to fully automate the process.

This study represents a continuation of our work presented in [19] where we show the historical analysis of the AUTOSAR meta-model evolution and [20] where we assess a number of metrics for monitoring the evolution of the AUTOSAR meta-model.

4. MEFIA METHOD DEFINITION

The goal of the *MeFiA* method is to identify optimal sets of new standardized architectural features to be adopted in the software development projects. In subsection 4.1 we define a meta-data model used for calculating the *Number of changes* (*NoC*) between different meta-model versions. In subsection 4.2 we define how to establish a link between the meta-model changes and the corresponding features. In subsection 4.3 we show how to search for the optimal sets of features to be adopted in the projects considering their prioritization and the required implementation effort in the meta-model based tools. Finally in subsection 4.4 we discuss our assumptions for the utilization of the *MeFiA* method.

4.1 Meta-data model for the changes

In order to calculate the number of changes between different meta-model versions, we use the meta-data model presented in Figure 1 [19]. This meta-data model represents a simplified version of the MOF meta-model [21].

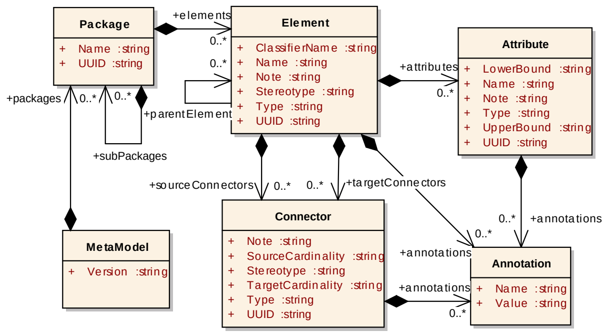


Figure 1: Meta-data model used calculating *NoC*

Meta-models are divided into *Packages* which contain *Elements* - classifiers and instances. Classifier *Elements* contain *Attributes*, *Connectors* of different *Type* (e.g., *Associations*, *Generalizations*) and *Annotations* describing their additional properties (e.g., regular expressions for strings). Instance *Elements* contain *Connectors* (except of *Type Generalization* as they represent concrete instances) and *Annotations* (e.g., C type, multiplicities). Finally, *Connectors* and *Attributes* can also contain *Annotations*. Each of the mentioned meta-elements of the meta-data model may contain additional properties captured in the attributes of the meta-elements such as *Name*, *Note* etc.

In order to compare different meta-model versions based on the meta-data model, we define the *Number of changes* (*NoC*) metric which counts the total number of relevant (causing re-work in the AUTOSAR-based tools) changes between two meta-model versions [19].

We define a 'change' as an atomic modification, addition and removal of the meta-data model elements and their properties (e.g. *Name*, *Note*). For example if one *Attribute* changed both its *Name* and its *Type*, this counts as two changes. Additionally when introducing or removing meta-data elements (e.g. *Attributes*) containing other meta-data elements (e.g. *Annotations*), the total number of changes considers both changes to the containing and contained meta-data elements (i.e. both *Attributes* and *Annotations*).

As an example, consider the introduction of one *Attribute* containing three *Annotations*. This counts as four changes - one for the *Attribute* and three for the *Annotations*. This way of calculating the total number of changes is justified by the fact that introduction of one *Element* cannot be counted as one change, like for example the introduction of an *Annotation*, as it requires higher implementation effort in the meta-model based tools.

To identify meta-elements in different meta-model versions, we used their *UUIDs* (Universally Unique IDentifiers of the objects) except for the *Annotations* where we used their *Name* as it is able to uniquely identify them in the context of one meta-data element.

4.2 Linking meta-model changes to features

In order to link meta-model changes to specific architectural features, certain process for implementing the changes in the meta-model needs to be established. This is specially the case with standardized features where many different parties may be involved in the definition of the final solution. The process we utilize in this paper relies on the existence of two commonly used tools in the distributed software development - a change management tool (e.g. Bugzilla, Jira) and a software repository (e.g. SVN, Git).

Change management tools such as Bugzilla can be used for documentation and traceability of the new standardized architectural features incorporated into different releases of the standard. Software repositories on the other hand can be used for documentation and traceability of different versions of the architectural meta-models between different releases of the standard. For linking the meta-model changes to the features, a link between these two tools can be established by following a process depicted in figure 2.

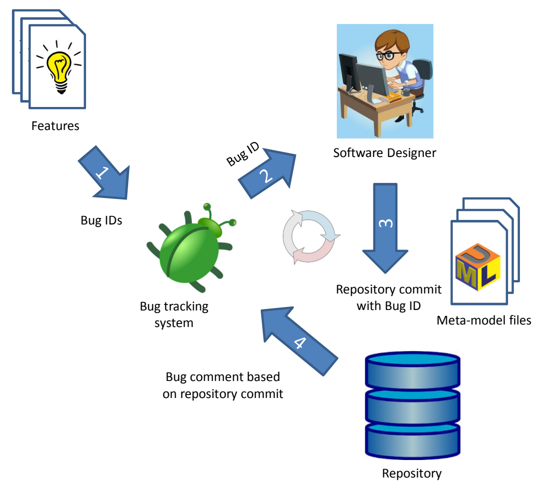


Figure 2: Linking meta-model changes to features

For each new feature to be implemented in the meta-model, an entry in the change management tool shall be created with a unique identifier (1). Software designer implementing the changes in the standardized meta-model shall use this identifier (2) in the commit message when committing a new version of the meta-model to the software repository (3). The process of modifying the meta-model and committing a new version to the software repository related to the same feature can be repetitive. Every time a commit to the repository is made, a comment is added to the bug with the identifier from the commit message (4). To assure that no links are omitted by the change implementers, the software repository should be configured to accept only certain structure of commit messages, e.g. a regular expression starting with the unique identifier of the entry in the change management tool (e.g. #12345).

4.3 Optimizing the set of adopted features

The search for the optimal sets of new standardized features to be adopted in the development projects is a multi-objective optimization problem [22, 23] with two objectives:

- Maximize the weighted number of features to be adopted based on their priority.
- Minimize the effort needed to implement the changes in the meta-model based tools based on the number of changes in the meta-model.

As the number of new features in different releases of the standard are limited to a reasonably high number and also considering the fact that the execution time is not critical, exhaustive algorithm which considers all possible combinations of features (i.e. starting with feature 1 only and ending with all features) is the most suitable algorithm.

For representing different solutions, a bit string s of length equal to the total number of new features n can be used as shown in formula 1.

$$s = (s_1, s_2, ..., s_n) \tag{1}$$

Each bit s_i in the bit string corresponds to one feature f_i (s_1 to feature f_1, etc.) and the value of the bit indicates whether the corresponding feature is included in the solution (value 1) or not (value 0). The solutions represent all possible combinations of bit values in the bit string which yields 2 to the power of n (2^n) different solutions.

In order to calculate the weighted number of features for each solution, a weight factor on an interval scale of 1 to 5 shall be assigned by the system designer to each feature where 5 is considered as the most important[2]. Then the total $wNoF$ for one solution represents the sum of the weights w_i of all features included in this solution, as shown formula 2.

$$wNoF(s) = \sum_{i=1}^{n} w_i * s_i \tag{2}$$

In order to measure the effort needed to adopt the features from one solution, we calculate the total NoC for this solution as the sum of the NoC for each feature included in the solution, as shown in formula 3.

$$NoC(s) = \sum_{i=1}^{n} NoC(f_i) * s_i \tag{3}$$

[2]We consider a scale of 1-5 to be the optimal but different scales could be more suitable for other situations.

This formula is based on the assumption that each change in the meta-model requires certain implementation effort in the meta-model based tools. Therefore the more changes we have, the more effort is needed to adopt the features causing these changes.

In order to represent all possible solutions and identify the optimal ones with respect to their *wNoF* (objective 1) and *NoC* (objective 2), a Pareto optimality chart presented in figure 3 can be used.

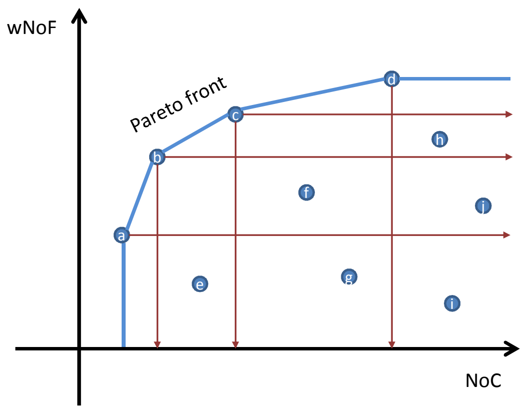


Figure 3: Pareto optimality chart

On the x axis we present the *NoC* and on the y axis we present the *wNoF* for all solutions. Then the solutions lying in the top and left most part of the chart form a *Pareto front*. These solutions are considered as better solutions, with respect to the two objectives, than the solutions lying below and to the right of them. For example solution b has both higher *wNoF* and lower *NoC* than solutions e, f, g, i and j. Therefore the optimal solution shall be discussed among the solutions on the *Pareto front*. This may significantly reduce potentially high number of solution and as such facilitate the decision making process.

Additionally, one solution shall be excluded from the consideration in one of the following 3 cases:

1. If there is a dependency between two or more features (i.e. one feature cannot be implemented without another feature) and a solution contains only a subset of the dependent features.

2. In case there is no need for some features and a solution contains at least one of them.

3. In case some features are required to be implemented and a solution does not contain all of them.

4.4 Assumptions for the MeFiA method

The *MeFiA* method is designed and assessed in the automotive domain where software development is done following the AUTOSAR standard. However we think that the applicability of this method could be extended to other domains developing software architectures based on a standard such as avionics based on IMA (Integrated Modular Avionics) [24], or banking based on BIAN (Banking Industry Architecture Network) [25]. In particular, the *MeFiA* method could be valuable for companies where:

1. The development of the architectural models is done based on a standardized meta-model which defines the syntax and the semantics for the models and serves as a basis for the implementation of the modeling tools.

2. The architectural models are exchanged between a number of actors involved in the development process.

3. A number of actors is involved in the development of the standardized meta-model.

The first point implies that the adoption of the new meta-model versions is needed to enable innovation in the development projects. The second point implies that the adoption of the new meta-model versions may potentially cause interoperability issues between the tools of different actors in the development process. The third point implies that possibly many new features driven by different stakeholders may be incorporated into the new versions of the standardized meta-model which requires careful cost-benefit analysis of which new features shall be selected for implementation.

In order to utilize the *MeFiA* method in other domains, the development of the standardized meta-model needs to satisfy the following conditions:

1. Standardized features shall be stored in a change management system, e.g. defect management system such as Bugzilla or Jira.

2. Different meta-model version shall be stored in a software repository such as SVN or Git and each meta-model commit shall be linked to the corresponding entry in the change management system.

3. Changes related to different features should not be committed simultaneously as the proposed method is not able to automatically detect which changes are related to which features. If there are cases like this, the changes in these commits shall be analyzed by the system designers and assigned to the right feature.

Additionally the *NoC* metric assumes that all changes have equal weight, i.e. that they all require similar implementation effort in the meta-model based tools. However changes introducing or breaking dependencies may require more effort. This could be improved in future work by classifying the changes into different types automatically detectable and assigning a weight to each type.

5. AUTOSAR BASED DEVELOPMENT OF AUTOMOTIVE SOFTWARE SYSTEMS

This section describes the development of the automotive software systems following the AUTOSAR standard which we used for the evaluation of the *MeFiA* method.

The development of the automotive software systems is distributed as they are developed in a collaborative environment which involves a number of actors. On the one side we have car manufacturers (OEMs) responsible for designing and verifying the architecture of the system. On the other side we have several layers of suppliers (e.g. application software suppliers, tool suppliers, hardware suppliers) responsible for the design, implementation and verification of the specific architectural components of the system [26]. As each party in the development process may use they own tools for working with the architectural models, the exchange of these models between different actors is quite challenging.

In order to facilitate this distributed development of the automotive software systems, AUTOSAR standard was introduced [27] as a joint partnership of the OEMs and their suppliers on the European market and wider. One of the main goals of AUTOSAR is to clearly separate the responsibility of different actors in the development process. For this purpose, a 3-layer software architecture [28] has been developed where the application software (i.e. vehicle functions such as auto-braking when pedestrians are detected in front of the car) is clearly separated from the underlying basic software (e.g. communication between ECUs, diagnostic services, etc.) and hardware [29].

Based on this architecture, AUTOSAR standardizes the exchange format for the architectural models. This is done by defining a meta-model which specifies the syntax and the semantics of the automotive modeling environment [30, 31] and serves as a basis for the development of the AUTOSAR based tools used for the modeling of the architectural components (e.g. application software components or basic software modules) of the system. In order to assure the interoperability between different AUTOSAR based tools, the exchanged architectural models needs to be fully compliant to the AUTOSAR meta-model. Figure 4 shows a simplified example of the usage of the AUTOSAR meta-model to allocate software components onto different ECUs.

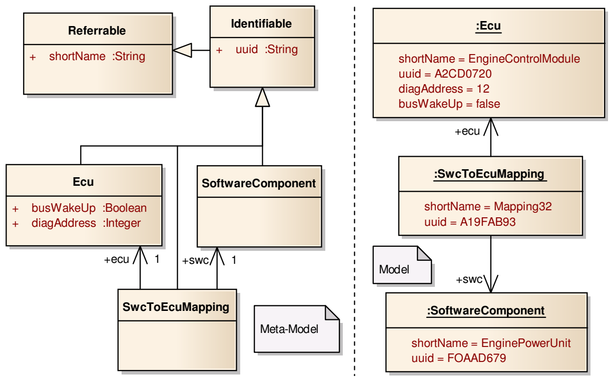


Figure 4: AUTOSAR Meta-Model example

The meta-model to the left defines how to allocate software components onto ECUs while the model to the right instantiates this meta-model by mapping the actual *EnginePowerUnit* software components onto *EngineControlModule* ECU. Software components and ECUs represents one of the main architectural units of the automotive software system and their allocation onto ECUs is an architectural feature. Another example of the architectural feature may be the use of Ethernet electronic bus as a communication medium between ECUs.

The architectural models are usually expressed in XML [32] and they are delivered by OEMs to the suppliers to continue with the implementation of the software, e.g. by developing behavioral models in tools such as *Matlab Simulink*. Before importing the models into the AUTOSAR based tools, they are validated by the AUTOSAR XML schema [33, 34] which is generated from the AUTOSAR meta-model (see [35] for more details about generating XML schema from the UML model). This process is depicted in Figure 5.

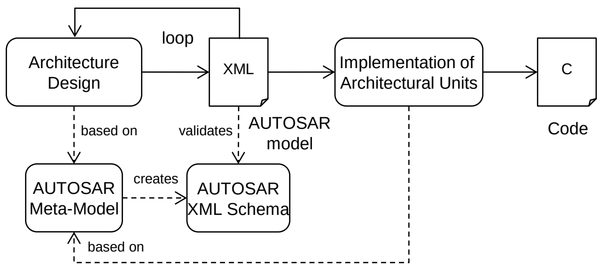


Figure 5: AUTOSAR based development process

To track the changes to the AUTOSAR specifications including the AUTOSAR meta-model, Bugzilla tool is used. Each change can be classified as clarification, correction or a new feature. For the new features which influence several different parts of the AUTOSAR architecture, new concepts are created and elaborated by different experts. The incorporation of the new concepts into the AUTOSAR releases is also documented in Bugzilla where a separate implementation task is created for the AUTOSAR meta-model.

For the development of the AUTOSAR specifications and the AUTOSAR meta-model, SVN repository is used. When updating the meta-model, the corresponding implementation task from the AUTOSAR Bugzilla needs to be referenced at the beginning of the SVN commit message (e.g. #54321). This reference is required by the tooling in order to avoid having SVN commits which are not linked to any Bugzilla entries. This enables full traceability of the changes to the AUTOSAR meta-model and other specifications and Bugzilla entries where the requests for these changes are described and elaborated.

6. THE RESULTS OF APPLYING MEFIA ON AUTOSAR FEATURES

We apply the *MeFiA* method on a set of 14 new features (referred to as concepts in AUTOSAR) incorporated into the AUTOSAR release *4.2.1*. In order to fully automate the generation of the optimal solutions, we implemented a tool which is used for this purpose at Volvo Cars. A brief description of each feature is presented below (the features have no dependencies between each other as stated in the feature documents of AUTOSAR):

- **Feat 1: Ethernet Switch Configuration** - provides means to configure Ethernet switches in an ECU.
- **Feat 2: Sender-Receiver Serialization** - significantly reduces the number of signals needed for the transmission of complex data.
- **Feat 3: CAN FD** - introduces a new communication protocol for CAN bus with higher bandwidth.
- **Feat 4: Efficient COM for Large Data** - faster transmission of large data through the ECU.
- **Feat 5: End-to-End Extension** - extends the safety communication means between the ECUs for the transmission of large data via TCP/IP.
- **Feat 6: Global Time Synchronization** - provides a common time base for accurate ECU data correlation.

- **Feat 7: Support for Post-Build ECU Configuration** - enables simultaneous configuration of ECU variants in one vehicle and different car lines.
- **Feat 8: Secure On-Board Communication** - provides mechanisms for securing the communication between the vehicle and the outside world.
- **Feat 9: Safety Extensions** - provides mechanisms to realize and document functional safety of AUTOSAR systems (e.g. according to the ISO 26262).
- **Feat 10: Decentralized Configuration** - provides means for transferring diagnostic needs of the OEMs to the suppliers.
- **Feat 11: Integration of Non-AUTOSAR Systems** - enables integration of non-AUTOSAR (e.g. Genivi) systems into AUTOSAR during development.
- **Feat 12: Efficient Non-Volatile Data Handling** - provides efficient mechanisms for software components to handle non-volatile data.
- **Feat 13: ECU State Manager Enhancement for Multi-Core** - provides support for state handling on multi-core ECUs.
- **Feat 14: ASIL-QM protection** - provides means to protect modules developed according to safety regulations from other modules.

In order to calculate the number of changes caused by each feature, we analyzed the changes between 97 SVN commits of the AUTOSAR meta-model. Out of these commits, 80 referred to only one feature and 17 additionally referred to other implementation tasks (13 to defects and 4 to other features). We excluded the changes not related to particular features from the latter 17 commits by analyzing them together with the AUTOSAR team at Volvo Cars.

As different companies may be interested in different features where some are required to be implemented, some are not needed and some may be considered only in case the cost of their implementation is acceptable, different scenarios for the usage of the MeFiA method are possible. For the purpose of this paper, we define the following scenario: A company wants to implement *Feat 1 (Ethernet Switch Configuration)* and *Feat 3 (CanFD)* and all other features are subject to the cost-benefit analysis with equal weights[3].

Subsection 6.1 discusses the optimal sets of features for the given scenario based on the analysis of their impact on the entire AUTOSAR meta-model. As the evolution of industrial meta-models may have significantly different impact on different parts of the AUTOSAR meta-model affecting different roles (teams) [19], the impact of the changes on different roles shall also be considered. This is especially important when analyzing the impact of feature related changes as some features may be related only to a limited number of roles. Subsection 6.2 shows how to search for the optimized set of features considering their impacts on a particular role and subsection 6.3 presents an example of how to aggregate the results of the analysis for different roles and use them in the decision making process.

[3]For simplicity and having in mind that different companies may use different prioritization of features, we assigned weight '1' to all analyzed features.

6.1 Optimization for the entire meta-model

Before searching for the optimal set of features to be adopted in the development projects, we should analyze if there are features which do not affect the AUTOSAR meta-model or features causing significantly more meta-model changes than the others. The adoption of these outliers should be analyzed separately in order not to cause major dissbalance in the results. In order to identify these kinds of features, a chart presenting the total number of meta-model changes for each feature can be used. Figure 6 shows the results of the *NoC* metric calculated for the 14 new features of the AUTOSAR release *4.2.1*.

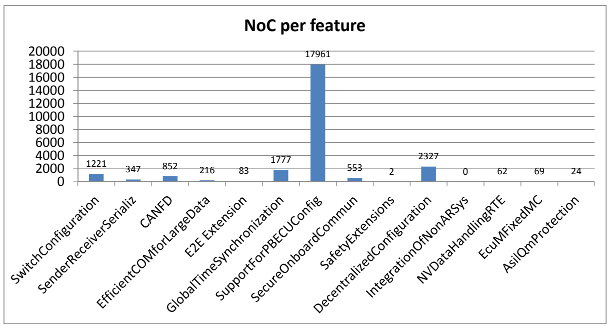


Figure 6: *NoC* per feature

We can see that the results of the *NoC* metric are very diverse ranging from *0* in case of *Feat 11 (Integration of Non-AUTOSAR Systems)* to *17961* in case of *Feat 7 (Support for Post-Build ECU Configuration)*[4]. These two features represent the outliers so we decided to exclude them from the analysis of optimal features with the recommendation to do the cost-benefit analysis for their adoption separately (e.g. *Feat 7* should probably not be selected due to its high number of changes affecting the AUTOSAR meta-model).

Additionally based on the scenario presented above, *Feat 1 (Ethernet Switch Configuration)* and *Feat 3 (CanFD)* shall be added to all solution as they are required to be implemented. Figure 7 shows the Pareto optimality chart with Pareto front containing 11 optimal solutions (*s1 - s11*) based on their impact on the entire AUTOSAR meta-model. Table 1 shows which features are included in which solution.

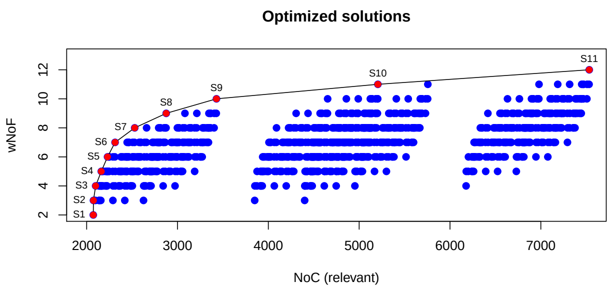


Figure 7: Optimal sets of features based on their impact on the entire meta-model

We can see that a higher increase in the number of changes needed to be implemented to support an additional features

[4]The reason why *Feat 11* brings no changes is that it does not affect the AUTOSAR based tools.

Table 1: Features of the optimal solutions

Solutions	Features
s1	Feat 1, 3
s2	Feat 1, 3, 9
s3	Feat 1, 3, 9, 14
s4	Feat 1, 3, 9, 14, 12
s5	Feat 1, 3, 9, 14, 12, 13
s6	Feat 1, 3, 9, 14, 12, 13, 5
s7	Feat 1, 3, 9, 14, 12, 13, 5, 4
s8	Feat 1, 3, 9, 14, 12, 13, 5, 4, 2
s9	Feat 1, 3, 9, 14, 12, 13, 5, 4, 2, 8
s10	Feat 1, 3, 9, 14, 12, 13, 5, 4, 2, 8, 6
s11	Feat 1, 3, 9, 14, 12, 13, 5, 4, 2, 8, 6, 10

starts with solutions *s7* and *s8*. Therefore the optimal solution should be searched among solutions *s6* and *s7*, i.e. *Feat 1 (Ethernet Switch Configuration)*, *Feat 3 (CAN FD)*, *Feat 9 (Safety Extensions)*, *Feat 14 (ASIL-QM protection)*, *Feat 12 (Efficient Non-Volatile Data Handling)*, *Feat 13 (ECU State Manager Enhancement for Multi-Core)*, *Feat 5 (End-to-End Extension)* and alternatively *Feat 4 E(fficient COM for Large Data)* shall be selected for adoption. However before making a final decision, the analysis of the impact of the features on different roles shall be done first, especially for the most critical ones.

6.2 Role-based optimization

For the role based analysis of the AUTOSAR meta-model changes, we consider 7 major roles in the AUTOSAR based automotive software development process which we defined in [19] (a mapping of roles to different parts of the AUTOSAR meta-model can also be found in this paper). A brief summery of these roles is presented below:

- **Role 1: Application software designers** - a team responsible for designing vehicle functions by defining software components and their interaction.
- **Role 2: ECU communication designers** - a team responsible for designing the communication between ECUs (e.g., transmitting signals on buses).
- **Role 3: ECU basic software configurators** - a team responsible for specifying the basic software configuration (i.e. which parameters are needed).
- **Role 4: Basic software designers** - a team responsible for designing the basic software modules (e.g., interfaces, services, etc.).
- **Role 5: ECU communication configurators** - a team responsible for configuring ECU communication basic software modules.
- **Role 6: Diagnostics configurators** - a team responsible for configuring basic software modules related to car diagnostics.
- **Role 7: Upstream mapping tool developers** - a team responsible for automated derivation of parts of the ECU configuration from the system models.

In order to identify the most critical roles, we analyze the chart presented in figure 8 which shows the number of feature related changes affecting each role separately (there may be other non-feature related changes affecting these roles such as corrections of defects).

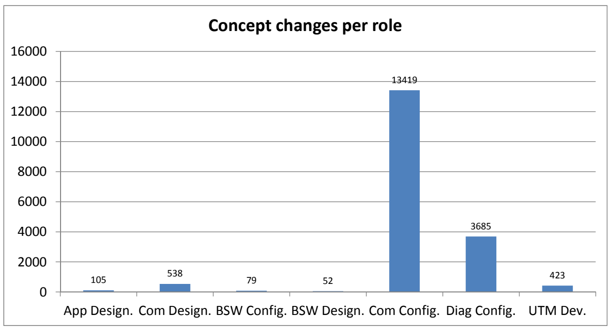


Figure 8: Number of feature changes per role

This conforms to our conclusion in paper [19] that the evolution of the AUTOSAR meta-model mostly affects the *Role 5 (ECU communication configurators)* and the *Role 6 (Diagnostics configurators)* while the other roles are less affected. Low impact is especially important for the *Role 1 (Application software designers)* and the *Role 2 (ECU communication designers)* as the architectural models developed by these two roles are usually exchanged between the OEMs and the suppliers and therefore they affect multiple actors in the development process. Considering this and the fact that the role of *ECU communication designers* is more than 5 times more affected by the changes than the role of *Application software designers*, we consider the *ECU communication designers* as the most critical role.

Based on the scenario presented above, Figure 9 shows the Pareto optimality chart with Pareto front containing 9 optimal solutions (*s12 - s20*) based on their impact on the *ECU communication designers* role. Table 2 shows which features are included in which solution.

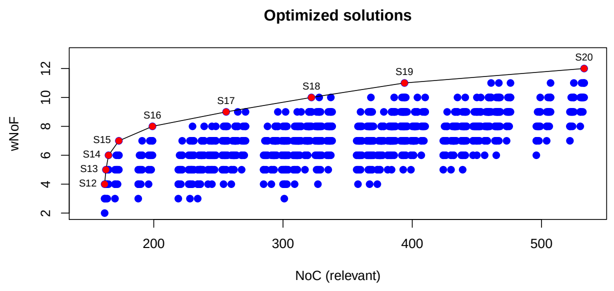


Figure 9: Optimal sets of features based on their impact on the ECU communication designers role

If we compare solutions *s12* and *s15*, we can see that with a relatively small increase in the number of changes we can implement 3 additional features - *Feat 4 (Efficient COM for Large Data)*, *Feat 9 (Safety Extensions)* and *Feat 12 (Efficient Non-Volatile Data Handling)*. We can also see that a big increase in the number of changes required to be implemented for adopting an additional feature starts with the implementation of the solution *s17*, in particular with the *Feat 5 (End-to-End Extension)*. Therefore the optimal solution should be searched among solutions *s15* and *s16*.

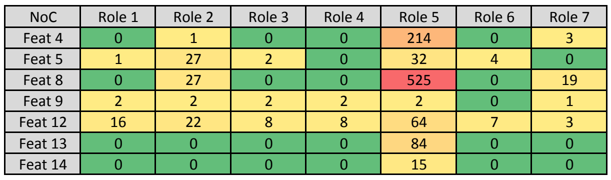
Table 2: Features of the optimal solutions for the ECU communication designers role

Solutions	Features
s12	Feat 1, 3, 13, 14
s13	Feat 1, 3, 13, 14, 4
s14	Feat 1, 3, 13, 14, 4, 9
s15	Feat 1, 3, 13, 14, 4, 9, 12
s16	Feat 1, 3, 13, 14, 4, 9, 12, 8
s17	Feat 1, 3, 13, 14, 4, 9, 12, 8, 5
s18	Feat 1, 3, 13, 14, 4, 9, 12, 8, 5, 10
s19	Feat 1, 3, 13, 14, 4, 9, 12, 8, 5, 10, 6
s20	Feat 1, 3, 13, 14, 4, 9, 12, 8, 5, 10, 6, 2

6.3 Aggregated role-based optimization

In the previous two subsections we discussed several optimal solutions based on the impact of their features on the entire meta-model (*s6* and *s7*) and the *ECU communication designers* role which is considered the most critical (*s15*) and *s16*. This means that apart from *Feat 1 (Ethernet Switch Configuration)* and *Feat 3 (CAN FD)* which are required, the decision about the adoption of *Feat 4 (Efficient COM for Large Data)*, *Feat 5 (End-to-End Extension)*, *Feat 8 (Secure On-Board Communication)*, *Feat 9 (Safety Extensions)*, *Feat 12 (Efficient Non-Volatile Data Handling)*, *Feat 13 (ECU State Manager Enhancement for Multi-Core)* and *Feat 14 (ASIL-QM protection)* shall be made.

Before making a final decision about the adoption of these features, their impact on other roles in the development process shall be considered. Figure 10 shows the number of changes needed to be implemented by different roles to support each one of the listed features.



NoC	Role 1	Role 2	Role 3	Role 4	Role 5	Role 6	Role 7
Feat 4	0	1	0	0	214	0	3
Feat 5	1	27	2	0	32	4	0
Feat 8	0	27	0	0	525	0	19
Feat 9	2	2	2	2	2	0	1
Feat 12	16	22	8	8	64	7	3
Feat 13	0	0	0	0	84	0	0
Feat 14	0	0	0	0	15	0	0

Figure 10: The impact of the features on other roles

We can see that the impact of the considered features on the other roles is smaller than their impact on the *ECU communication designers* except for the *ECU communication configurators* role. Therefore the decision about which new features shall be adopted together with *Feat 1 (Ethernet Switch Configuration)* and *Feat 3 (CAN FD)* shall be made based on the assessment of their impact on the *ECU communication configurators* role.

7. CONCLUSION AND FUTURE WORK

Following a standard in the development of software architectures requires continuous adoption of the new standardized features where only their subset is required by individual companies. In order to decide upon which set of new features shall be adopted in the development projects, the assessment of their impact on the software tools used for modeling the architectural components is an important aspect in the decision making process. This is specially the case with large distributed systems where the architectural models are exchanged between a number of actors in the development process in order to assure the interoperability between different tools working with the models.

In order to facilitate the decision making process of which new standardized architectural features shall be implemented in the development projects, we defined a method - *MeFiA* - for identifying the optimal sets of features. The optimal sets are identified based on the tradeoff analysis between their importance for the final product and the amount of rework needed in the software modeling tools caused by the changes in the standardized meta-model.

We evaluate the *MeFiA* method in the automotive domain where software architectures are developed following the AUTOSAR standard. We present the cost-benefit analysis for adoption of 14 new architectural features of the AUTOSAR release *4.2.1*. The analysis is done based on a scenario considering different roles in the development process.

We concluded that the proposed method is applicable for identifying the optimal set of new architectural features to be adapted in the automotive development projects. However we also believe that the method is applicable to a wider range of domains where software is developed based on industrial meta-models and where the link between a change management tool and a software repository is maintained. Further empirical studies are needed to support this.

In our future work we intend to evaluate the robustness of the *MeFiA* method to violations of the method assumptions, in particular the assumption of equal weights of different meta-model changes. We plan to address this by classifying the changes into different categories where each category contains a predefined weight. We also plan to investigate the impact of different prioritization of features in a case study at different companies.

8. ACKNOWLEDGMENTS

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 Corporation for contributing to the work.

9. REFERENCES

[1] M. Di Natale and A. L. Sangiovanni-Vincentelli, "Moving From Federated to Integrated Architectures in Automotive: The Role of Standards, Methods and Tools," *Proceedings of the IEEE*, vol. 98, no. 4, pp. 603–620, 2010.

[2] 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.

[3] S. Fűrst, "Challenges in the Design of Automotive Software," in *Proceedings of the European Conference on Exhibition on Design, Automation & Test*, 2010, pp. 256–258.

[4] S. Gal-Oz, "Standard API Would Significantly Accelerate Embedded System Development," *Real-Time Magazine*, vol. 5, pp. 81–87, 1999.

[5] *Automotive Open System Architecture*, AUTOSAR, www.autosar.org, 2003.

[6] R. Kazman, M. Klein, M. Barbacci, and T. Longstaff, "The Architecture Tradeoff Analysis Method," in

Proceedings of the IEEE International Conference on Engineering of Complex Computer Systems, 1998, pp. 68–78.

[7] R. Nord, M. Barbacci, P. Clements, R. Kazman, M. Klein, L. O'Brien, and J. Tomayko, "Integrating the Architecture Tradeoff Analysis Method (ATAM) with the Cost Benefit Analysis Method (CBAM)," Software Engineering Institute, Tech. Rep., 2003.

[8] M. Asadi, S. Soltani, D. Gasevic, M. Hatala, and E. Bagheri, "Toward Automated Feature Model Configuration with Optimizing Non-Functional Requirements," *Journal of Information and Software Technology*, vol. 56, pp. 1144–1165, 2014.

[9] J. White, B. Dougherty, and D. Schmidt, "Selecting Highly Optimal Architectural Feature Sets with Filtered Cartesian Flattening," *Journal of Systems and Software*, vol. 82, pp. 1268–1284, 2009.

[10] A. Bachmann, C. Bird, F. Rahman, P. Devanbu, and A. Bernstein, "The Missing Links: Bugs and Bug-fix Commits," in *Proceedings of the ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2010, pp. 97–106.

[11] M. Fischer, M. Pinzger, and H. Gall, "Analyzing and Relating Bug Report Data for Feature Tracking," in *Proceedings of the Working Conference on Reverse Engineering*, 2003, pp. 90–100.

[12] H. Gustavsson and U. Eklund, "Architecting Automotive Product Lines: Industrial Practice," in *Proceedings of the International Conference on Software Product Lines: Going Beyond*, 2010, pp. 92–105.

[13] U. Eklund, C. M. Olsson, and M. Ljungblad, "Characterising Software Platforms from an Architectural Perspective," in *Proceedings of the European Conference on Software Architecture*, 2013, pp. 344–347.

[14] S. Dersten, J. Axelsson, and J. Frőberg, "Effect Analysis of the Introduction of AUTOSAR," in *Proceedings of the International Conference on Software Engineering and Advanced Applications*, 2011, pp. 239–246.

[15] H. Soubra and K. Chaaban, "Functional Size Measurement of Electronic Control Units Software Designed following the AUTOSAR Standard," in *Proceedings of the International Conference on Software Process and Product Measurement*, 2012, pp. 78–84.

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

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

[18] J. W. Creswell, *Research Design.* Sage, 1994.

[19] D. Durisic, M. Staron, M. Tichy, and J. Hansson, "Evolution of Long-Term Industrial Meta-Models - A Case Study of AUTOSAR," in *Proceedings of the Euromicro Conference on Software Engineering and Advanced Applications*, 2014, pp. 141–148.

[20] D. Durisic, M. Staron, and M. Tichy, "Quantifying Long-Term Evolution of Industrial Meta-Models - A Case Study," in *Proceedings of the International Conference on Software Process and Product Measurement*, 2014, pp. 104–113.

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

[22] J. Branke, K. Deb, K. Miettinen, and R. Slowiński, *Multiobjective Optimization.* Springer Berlin Heidelberg, 2008.

[23] M. Harman and B. Jones, "Search Based Software Engineering," *Journal of Information and Software Technology*, vol. 43, no. 14, p. 833âĂŞ839, 2001.

[24] C. Watkins and R. Walter, "Transitioning from Federated Avionics Architectures to Integrated Modular Avionics," in *Proceedings of the Conference on Digital Avionics Systems*, 2007, pp. 2.A.1–1 – 2.A.1–10.

[25] *Banking Industry Architecture Network*, BIAN, www.bian.org, 2008.

[26] B. Boss, "Architectural Aspects of Software Sharing and Standardization: AUTOSAR for Automotive Domain," in *Proceedings of the International Workshop on Software Engineering for Embedded Systems*, 2012, pp. 9–15.

[27] C. Wang, L. Ge, and T. Lee, "Automotive ECU Software Design Based on AUTOSAR," *Journal of Applied Mechanics and Materials*, vol. 577, pp. 1034–1037, 2014.

[28] C. Briciu, I. Filip, and F. Heininger, "A New Trend in Automotive Software: AUTOSAR Concept," in *Proceedings of the International Symposium on Applied Computational Intelligence and Informatics*, 2013, pp. 251–256.

[29] B. Huang, H. Dong, D. Wang, and G. Zhao, "Basic Concepts on AUTOSAR Development," in *Proceedings of the International Conference on Intelligent Computation Technology and Automation*, 2010, pp. 871–873.

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

[31] 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.

[32] J. Suzuki and Y. Yamamoto, "Managing the Software Design Documents with XML," in *Proceedings of the International Conference on Computer Documentation*, 1998, pp. 127–136.

[33] M. Pagel and M. Brőrkens, "Definition and Generation of Data Exchange Formats in AUTOSAR," in *Proceedings of the European Conference on Model Driven Architecture-Foundations and Applications*, 2006, pp. 52–65.

[34] U. Honekamp, "The Autosar XML Schema and Its Relevance for Autosar Tools," *IEEE Software*, vol. 26, pp. 73–76, 2009.

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