Stefan Kubica, Hagen Ringshausen,
Jörg Reiff-Stephan, Marius Schlingelhoff (Hrsg.)
1. Automobil Symposium Wildau: Tagungsband
Technische Hochschule Wildau 2016

Towards Reducing the Complexity of Enterprise Architectures by Identifying Standard Variants Using Variability Mining

Kenny Wehling, David Wille, Martin Pluchator, Ina Schaefer

doi: 10.15771/ASW_2016_6

Abstract

For decades, Enterprise Architectures (EAs) of car manufacturers have been constantly evolved to respond to growing requirements. As a consequence, EAs have often reached a very high level of complexity, which leads to problems in adapting EAs to new environmental conditions. Such a new condition is, for instance, digitalization of society (e.g., social media, Internet of Things) which has a huge effect on the automotive industry and the grown EA. Resulting changes in complex EAs have long implementation cycles, require enormous communication efforts, and lead to high development costs. To alleviate these problems, in this paper, we present a concept to reduce the complexity of grown EAs by adapting the Family Mining approach. This approach is originally used to compare block-oriented models, such as MATLAB/Simulink models, and to identify commonalities and differences between these models. In our concept, we utilize the Family Mining approach to analyze the variability of a particular EA and to identify the contained variants. All information about the variability and the variants will be used to derive standard variants representing default solutions for different issues. Using these standard variants, the existing EA will be restructured involving economic considerations (e.g., which standard variant yields best benefits under certain circumstances). Hence, applying this concept to a complex EA should allow reducing the complexity of the EA, alleviating related problems and making suitable design decisions for future extensions.

1. Introduction

For a long time, the automotive industry has worked on optimizing its manufacturing and customer processes. Thus, IT-Systems and whole Enterprise Architectures (EA) have been constantly evolved to respond to growing requirements. As a consequence, EAs of car manufacturers have often reached a very high level of complexity, which is reflected in thousands of components and corresponding relations as well as a vast heterogeneity. Such a grown EA allows to execute automotive processes with support of established IT-Systems, also called »classic IT«. Recently, the automotive industry has been influenced by digitalization of society, which has a huge impact on the grown EA and requires comprehensive adaptions.

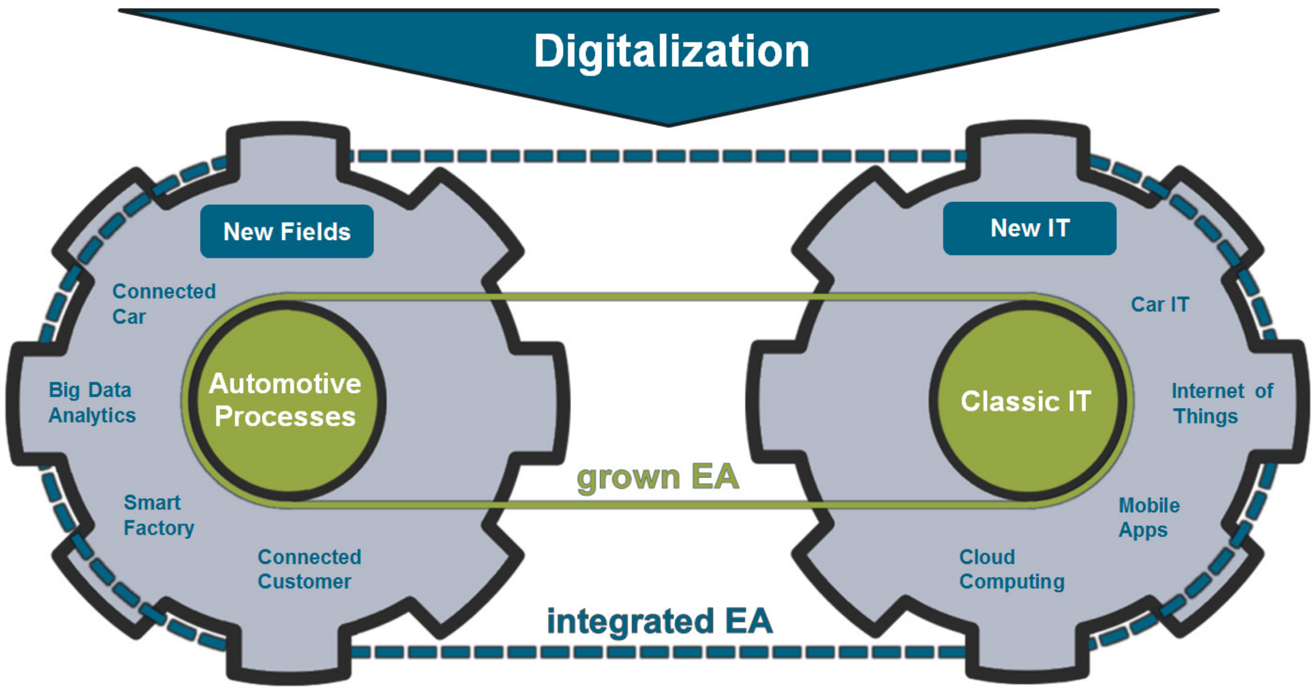

Fig. 1 The impact of digitalization on grown Enterprise Architectures of car manufacturers

Due to the innovating driver digitalization, *new IT* (e.g., Cloud Computing and Car IT) arise and *new fields* (e.g. Connected Car and Smart Factory) are available. Both, new IT and new fields, need to be successfully *integrated* in the grown EA to take advantage of digitalization. Fig. 1 shows this impact of digitalization on grown EAs of car manufacturers. Established automotive processes and classic IT are embedded in new fields and new IT. While the grown EA keeps automotive processes moving, only the integrated EA allows car manufacturers to utilize and benefit from digitalization.
Grown EAs of car manufacturers are not prepared for these enormous changes resulting from digitalization and it's new IT and new fields. The reasons are various and there are only listed a few below. Firstly, grown EAs of car manufacturers are too complex and too confusing in order to manage such large changes successfully. Secondly, there is often no sufficient documentation of the IT-Landscape, the used IT-Systems, and the relations between them, which leads to long implementation cycles, enormous communication efforts, and high development costs. Thirdly, artefacts for a specific context in grown EAs are rather implemented new then reused from an existing solution of another context. This ends up in increasing the complexity of a *grown EA* and makes it even more difficult to manage future changes.
In order to integrate the new IT and new fields into grown EAs of car manufacturers without impairing productive systems and loosing knowledge, the complexity of grown EAs has to be initially reduced to an appropriate level. For this purpose, we introduce a concept in this paper which has to be evaluated in future work. In particular, we make the following contributions:

- We propose an idea to automatically generate clusters based on a grown EA to detect applications with similar types of technical infrastructures.
- We automatically analyze the variability of detected clusters to emphasize commonalities and differences of related technical infrastructures.
- We identify standard variants to reduce the variability within detected clusters which is the foundation for restructuring a grown EA.

This paper is structured as follows: Section 2 provides background on EAs, complexity of EAs and Family Mining. Section 3 describes our concept to reduce complexity of EAs by Identifying standard variants using Family Mining. Section 4 discusses related work and Section 5 concludes with an outlook to future work.

2. Background

In Section 2.1, we provide a definition for EA followed by Section 2.2 which describes the complexity of EAs. In the last Section 2.3, we introduce family mining, an algorithm for variability mining.

2.1 Enterprise Architecture

The field of EA was first introduced by Zachmann in 1987. In his work he developed a multiperspective approach to information systems and their architecture (Zachman 1987). Similarly, Richardson et al. describe EA as a multidimensional view to information systems consisting of »interrelated data, hardware, software, and communi-

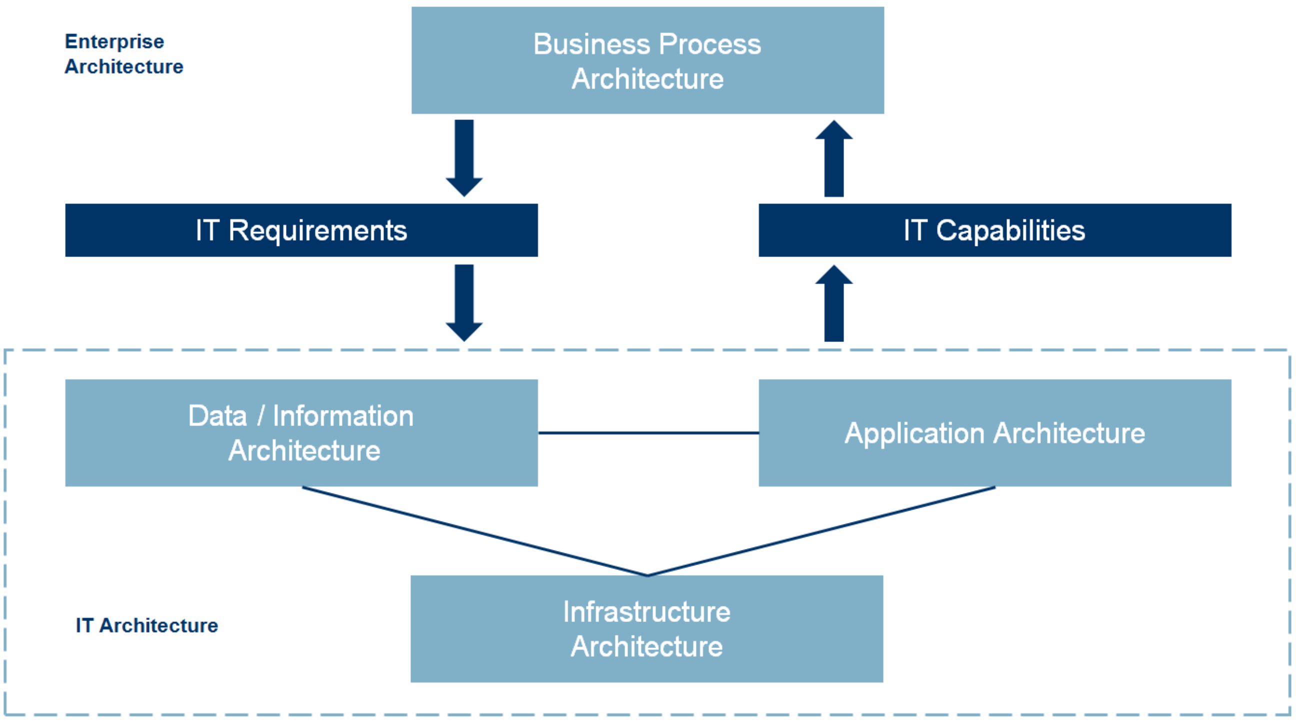

Fig. 2 The four dimensions of Enterprise Architecture (adapted from Schütz et al. 2013a)

cations« (Richardson et al. 1990). Furthermore, Ross extends the term EA with the dimension of the enterprise and its business processes (Ross 2003). Based on these preceding works, established EA frameworks, such as The Open Group Architecture Framework (TOGAF), focus on the four architecture dimensions *Business Process*, *Data/Information*, *Application*, and *Infrastructure* (see Fig. 2) (The Open Group 2011).
In this paper, we concentrate on the Infrastructure Architecture, but we are planning to extend our concept to the remaining dimensions of EA in future work.
From a system theoretical perspective, EA can be described as a system which consists of components and relations between them (IEEE Architecture Working Group 2000). These components and relations can be found in every dimension of EA. For instance, components in an Infrastructure Architecture might be hardware or network equipment for the technical foundation of an application.

2.2 Complexity of Enterprise Architectures

Several approaches have been developed to determine complexity of EAs. Schneider et al. present a good overview about available methods (Schneider et al. 2014). One of these approaches, which fits best to our concept, has been evolved by Schütz et al. (2013b) and is described below. Based on the system theoretical view on EA, Schütz et al. define Complexity (C) as a *tuple that consists of the number (N) and the heterogeneity (H) of components (T) and relations (R) embedded in an EA*:

$$C_Y = \left(N_Y, H_Y\right), with\ Y \in \{T, R\} \tag{1}$$

To determine the number of components or relations in an EA, it is only necessary to count them. For determination of heterogeneity, Schütz et al. propose the Entrophy Measure (EM) (see Equation 2). It contains the parameter

$$EM = -\sum_{i=1}^{n} p_i \ln\left(p_i\right) \tag{2}$$

p_i which states the relative frequency of characteristic i of a specific element considered in an EA. The parameter p_i can be defined as seen bellow involving the parameter x_i which quantifies the absolute number of elements assigned to characteristic i (see Equation 3).

$$p_i = \frac{x_i}{\sum_{j=1}^{n} x_j} \tag{3}$$

In Section 3, we give a motivating example that shows the determination of complexity and demonstrates our approach.

2.3 Family Mining

In previous work, we introduced family mining as a mining technique to automatically analyze the variability inherent in MATLAB/Simulink model variants (Holthusen et al. 2014, Wille 2014, Wille et al. 2013). By comparing a set of related models, this approach allows to identify mandatory parts (i.e., common to all variants), optional parts (i.e., only contained in particular variants), and alternative parts (i.e., mutually exclusive). The family mining algorithm stores the results in so-called 150 % models containing all artefacts from the variants together with their identified variability. Such 150 % models can be visualized by showing all contained elements in their standard notation together with visual elements denoting their variability. Mandatory elements are marked with an exclamation mark (i.e., !), optional elements are marked with a question mark (i.e., ?), and alternatives are marked with double arrow (i.e., ⇔).
We plan to adapt the ideas of our family mining approach in order to identify variability information in grown EAs on different assets. For instance, we want to analyze the variability information to identify standard variants for *technical architectures (TAs)* (i.e., a common hardware base). Another plan is to ascertain the variability of applications in different business locations to determine standard applications for equal business supports. Using these ideas our overall goal is to reduce the complexity of grown EAs and to alleviate relating problems.

3. Concept for Reducing the Complexity of Enterprise Architectures

Below, we show the determination of EA complexity by means of a motivating example which is also used for our further consideration in this paper. Tab. 1 describes three TAs from three different applications that represent a simple EA. Each TA consists of a typical Client-Server-Architecture which is separated in the three layers

Tab. 1 Motivating example to demonstrate our approach

	TA 1		TA 2		TA 3	
	Client	Server	Client	Server	Client	Server
Presentation	P_1	–	P_2	–	P_3	–
Application Server	–	A_1 A_2	–	A_1	–	A_1
Hardware & OS	–	H_1	–	H_2	–	H_3 H_4

Presentation, *Application Server*, and *Hardware & Operating System (OS)*. In this example, we focus on components and do not take relations into account, in order to demonstrate our concept on a less complex use case. In future work, we will also concentrate on these relations. Thus, complexity in our example can be determined by $C_T = (N_T, H_T)$ (cf. Section 2.2).
In Tab. 1, we added components to the layers by using the first letter of the corresponding layer and a number to distinguish between differing components. All TAs in our example are related due to the application server A_1

Tab. 2 Determination of complexity for our motivating example

	N_T	H_T
Presentation	3	0.64
Application Server	4	0.56
Hardware & OS	4	1.39
Average	**3.67**	**0.86**

which is installed in each TA.
As seen in Tab. 2, the average complexity of our exemplary EA is C = (3.67, 0.86). The lower the values for N_T and H_T are, the lower is the complexity of a given EA.
In the following, we want to introduce our concept to reduce the calculated complexity for a given EA. The first step is to identify clusters in a grown EA to distinguish between sets of related TAs. This is shown in Section 3.1 and can be considered as a pre-processing step in our example. The next step is to automatically analyze and identify the variability of the resulting clusters by the Family Mining methodology which is described in Section 3.2. Based on this, the next step is to determine standard variants within the detected clusters which is shown in Section 3.3. How to use the identified standard variants to reduce the variability of the detected clusters and, thus, to reduce the complexity of a grown EA, is explained in Section 3.4.

3.1 Clustering the Enterprise Architecture

To get valid information about the variability in the Infrastructure Architecture level of grown EAs, it is necessary to group TAs first in types of similar characteristic. Such a similar characteristic is determined by one or more components which are essential for that specific type. For instance, a group of TAs running a Java Application Server might be determined by the component IBM WebSphere Application Server. If all available TAs were analyzed at once without grouping, no commonalities would be detected, because grown EAs of car manufacturers consist of thousands of TAs and the enormous variability between them would be identified as optional elements. For example, a comparison of a TA for a classic client-server application with a TA for a web server would identify large differences and, thus, most parts would be identified as optional for a common TA base.

In our concept, we propose to cluster the set of available TAs of a grown EA during a pre-processing step prior executing the variability mining. The approach by Babur et al. (2016) is one possible solution to cluster the given TAs. Their approach uses a bigram vocabulary created from the compared TA models and allows to calculate the term incidence matrix for the compared models (i.e., to show the presence of bigrams in the models). Using statistical methods, such as the Manhattan distance (Krause 2012), the distance between the matrix vectors can be calculated to cluster related models. Our exemplary TAs have been clustered to a set of TAs which have the Application Server A_1 installed.

3.2 Mining the Variability of Detected Clusters

Once the clusters of TAs have been detected, the next step of our approach is to automatically identify the variability of each cluster using the Family Mining methodology by comparing the related TAs with each other.
After analyzing the exemplary cluster of our TAs, the following 150 % model has been created.
As we can see in Tab. 3, the 150 % model constitutes all information about commonalities and differences of the processed TAs from Tab. 1 in a single TA. At the Presentation Layer, the client has the component P_1 or P_2 installed. Both are alternative elements with the same functionality. At the Application Server layer, the server in each TA within this cluster has implemented the component A_1, thus, it is a mandatory element. In contrast, component A_1 is only optional as it solely appears in TA 1. At the lowest layer Hardware & OS, the three alternative components H_1, H_2, and H_3 are used for the server. Additionally, an optional component H_4 has been detected for the server on this level.

Tab. 3 150 % model of our motivating example

	TA (150 % model)	
	Client	Server
Presentation	$P_1 \Leftrightarrow P_2$	–
Application Server	–	$!A_1$ $?A_2$
Hardware & OS	–	$H_1 \Leftrightarrow H_2 \Leftrightarrow H_3$ $?H_4$

3.3 Identifying Standard Variants for Detected Clusters

The next step of our approach is to identify standard variants for each cluster. These standard variants might be one or more single standard components or an entire standard TA for a given cluster. By identifying and using these standard components within a cluster, it is possible to reduce the variability and, thus, the complexity of this cluster. In conclusion, each cluster with less variability options reduces the overall complexity of a grown EA. Standard variants, based on a 150 % model, can be identified by three different approaches:

- manual decisions by an expert
- semi-automatic decisions realized by a rule based system supported by an expert
- fully-automatic decisions realized by a rule based system containing comprehensive domain knowledge

For our motivating example, we identified standard variants using a manual approach as the other approaches still have to be developed which is planned for future work. As shown in Tab. 4, our standard variant for the Presentation layer is P_2, because it is equal in functionality to P_1 and is preferred. At the Application layer, the mandatory component A_1 has become a standard variant. In contrast, component A_2 has been deleted as it was only optional and not needed for component A_1. At the layer Hardware & OS, component H_4 has been erased for the same reason like component A_2. From the remaining components on this layer, H_1 and H_2 have been identified as alternative standards because of strategic reasons. Therefore, component H_3 is not supported anymore.

Tab. 4 Identified standard variants for our motivating example

	TA (150 % model)	
	Client	Server
Presentation	$!P_2$	–
Application Server	–	$!A_1$
Hardware & OS	–	$H_1 \Leftrightarrow H_2$

3.4 Restructuring Detected Clusters Using the Identified Standard Variants

To reach our overall target of complexity reduction in grown EAs, the identified standard variants need to be realized. Therefore, each TA in a detected cluster has to match the identified standard variants and, if necessary, has to be restructured.
The restructured TAs of our exemplary cluster can be seen in Tab. 5. Each TA implements the standard component P_2 at Presentation layer and the standard component A_1 at Application layer. At Hardware & OS layer, only TA 3 had to be restructured and now uses standard component H_1.
After a successful restructuring, we have determined the complexity of our exemplary EA once again. The results are shown in Tab. 6 compared to the results prior restructuring.

As we can see in Tab. 6, the complexity in our example has been reduced at each single layer by restructuring the related TAs using the identified standard variants. Thus, the average complexity of our exemplary EA has also been reduced from $C_1 = (3.67, 0.86)$ to $C_2 = (3, 0.21)$ resulting in a percental reduction of $\Delta C = (-18.26\,\%, -75.58\,\%)$.

4. Related Work

In literature, there are several works regarding design principles for planning the to-be-architecture of EAs (i.e., Greefhorst & Proper 2011, Haki & Legner 2013, Richardson et al. 1990). As these approaches only consider to-be-architectures from a planning perspective and do not take complexity of as-is-architectures into account, they are not suitable for us. Additionally, Schütz et al. (2013a) propose design principles which also include consideration of complexity. Their design principles are based on the introduced definition of complexity (see Section 2.3) and have been evaluated by a project applied the action design research (ADR) method. In conclusion, Schütz et al. have extended their approach to a set of seven design principles which include, for instance, consideration of end-user-acceptance and data quality. Their definition of design principles is very interesting for identifying standard variants in our approach and will be considered in future work. However, our approach focuses more on automatically analyzing the variability of grown EAs and giving specific recommendations for restructuring of these EAs to reduce their overall complexity.

5. Conclusion and future work

In this paper, we introduced a concept to reduce the complexity of grown EAs by identifying standard variants and using them to restructure a grown EA. We have shown that a grown EA first has to be clustered in similar types of TAs and afterwards has to be analyzed by the Family Mining methodology in order to identify the variability of detected clusters. Based on the variability information of a 150 % model, showing commonalities and differences of related TAs within a detected cluster, standard variants can be identified for each cluster. These standard variants are used to restructure the detected clusters and, thus, the grown EA. After a successful restructuring, a reduced complexity of the grown EA can be determined.
In all cases of our motivating example, reduction of complexity has been able to be achieved by either declaring a non-mandatory element as a standard variant or erasing it. In conclusion, identifying standard variants is one of the most important steps in our approach and should be executed with highest possible accuracy. Not only domain knowledge has to be taken into account, but also strategic and economic considerations as well as potential costs for restructuring a grown EA. Further-

Tab. 5 Motivating example with standard variants

	TA 1		TA 2		TA 3	
	Client	Server	Client	Server	Client	Server
Presentation	P_2	–	P_2	–	P_2	–
Application Server	–	A_1	–	A_1	–	A_1
Hardware & OS	–	H_1	–	H_2	–	H_1

Tab. 6 Determination of complexity for our motivating example prior and after restructuring

	prior restructuring (C_1)		after restructuring (C_2)	
	N_T	H_T	N_T	H_T
Presentation	3	0.64	3	0
Application Server	4	0.56	3	0
Hardware & OS	4	1.39	3	0.64
Average	**3.67**	**0.86**	**3**	**0.21**

more, our motivating example consists of only three TAs. In contrast, grown EAs of car manufacturers consist of thousands of different TAs. Hence, our approach needs to be evaluated with significantly larger EAs from real world scenarios.

In future work, we plan to evaluate our approach under more realistic circumstances with larger EAs from industrial contexts. Furthermore, we want to extend our approach to relations between components and to the remaining levels of EA. Additionally, we plan to create an appropriate methodology to identify standard variants considering the mentioned aspects and design principles. For this methodology, we also plan to develop a semi-automatical solution.

References

Babur Ö, Cleophas L, Verhoeff T, van den Brand M (2016) Towards Statistical Comparison and Analysis of Models. In: Proc 4th Int Conf Model-Driven Engineering and Software Development (MODELSWARD), 19–21 Feb 2016, Rome, p 84

Greefhorst D, Proper E (2011) Architecture Principles. The Cornerstones of Enterprise Architecture. The Enterprise Engineering Series, vol 4. Springer, Berlin, Heidelberg. ISBN: 978-3-642-20278-0. doi: 10.1007/978-3-642-20279-7

Haki MK, Legner C (2013) Enterprise Architecture Principles In Research And Practice: Insights From An Exploratory Analysis. In: Proc 21st Europ Conf Information Systems (ECIS) Compl Res, 5–8 Jun 2013, Utrecht, p 204

Holthusen S, Wille D, Legat C, Beddig S, Schaefer I, Vogel-Heuser B (2014) Family Model Mining for Function Block Diagrams in Automation Software. In: Proc 18th Int Software Product Line Conf (SPLC): 2nd Int Workshop Reverse Variability Engineering (REVE), 15–19 Sep 2014, Florence. ACM, ISBN: 978-1-4503-2739-8, pp 36–43. doi: 10.1145/2647908.2655965

IEEE Architecture Working Group (2000) IEEE Recommended Practice for Architectural Description of Software-Intensive Systems. IEEE Standard 1471-2000. IEEE, Piscataway. ISBN: 978-0-7381-2518-3. doi: 10.1109/IEEESTD.2000.91944

Krause EF (2012) Taxicab Geometry. An Adventure in Non-Euclidean Geometry. Dover Books on Mathematics. Dover, Newburyport. ISBN: 978-0-486-25202-5

Richardson GL, Jackson BM, Dickson GW (1990) A Principles-Based Enterprise Architecture. Lessons from Texaco and Star Enterprise. MIS Quarterly 14(4):385. doi: 10.2307/249787

Ross JW (2003) Creating a Strategic IT Architecture Competency: Learning in Stages. MIT Sloan Working Paper 4314-03, Center for Information Systems Research Working Paper 335. SSRN Electron J. doi: 10.2139/ssrn.416180

Schneider AW, Zec M, Matthes F (2014) Adopting Notions of Complexity for Enterprise Architecture Management. In: 20th Americas Conf Information System (AMCIS), 07–09 Aug 2014, Savannah, ISBN: 978-0-692-25320-5

Schütz A, Widjaja T, Gregory R (2013a) Escape from Winchester Mansion – Toward a Set of Design Principles to Master Complexity in IT Architectures. In: Proc 34th Int Conf Information Systems ICIS, 15–18 Dec 2013, Milan, ISBN: 978-0-615-93383-2

Schütz A, Widjaja T, Kaiser J (2013b) Complexity in Enterprise Architectures – Conceptualization and Introduction of a Measure from a System Theoretic Perspective. In: Proc 21st Europ Conf Information Systems (ECIS) Compl Res, 5–8 Jun 2013, Utrecht, p 202

The Open Group (2011) Welcome to TOGAF® Version 9.1, an Open Group Standard. http://pubs.opengroup.org/architecture/togaf9-doc/arch. Accessed 16 Feb 2016

Wille D (2014) Managing Lots of Models: The FaMine Approach. In: Proc 22nd ACM SIGSOFT Int Symp Foundations of Software Engineering (FSE), 16–21 Nov 2014, Hong Kong, pp 817–819. doi: 10.1145/2635868.2661681

Wille D, Holthusen S, Schulze S, Schaefer I (2013) Interface Variability in Family Model Mining. In: Proc 17th Int Software Product Line Conf (SPLC): Proc Int Workshop Model-Driven Approaches in Software Product Line Engineering (MAPLE), 26–30 Aug 2013, Tokyo. ACM, ISBN: 978-1-4503-2325-3, pp 44–51. doi: 10.1145/2499777.2500708

Zachman JA (1987) A framework for information systems architecture. IBM Syst J 26(3):276–292. doi: 10.1147/sj.263.0276

Autors

Kenny Wehling
Volkswagen AG
kenny.wehling@volkswagen.de

David Wille
TU Braunschweig
d.wille@tu-bs.de

Martin Pluchator
Volkswagen AG
martin.pluchator@volkswagen.de

Ina Schaefer
TU Braunschweig
i.schaefer@tu-bs.de